Interaktion

INTERAKTION: Users Group for the Interak Computer.

Newsletter Numbers 3 and 4 June and September 1983

CONTENTS

The Interaktion Newsletter is Published and Edited by P.P. Vella,
19 Ford Drive, Yarnfield, Nr. Stone, Staffs. ST15 ORP

Telephone 0785 760 250.

Editor's Letter

Thank you everybody for the contributions and suggestions which keep rolling in - when I came back from holiday I found 40 letters waiting for me from users and they're still coming, so I hope you will bear with me if I haven't been able to answer yours yet. Since our last newsletter the number of members have tripled, so you can see the scale of the problem. As the planned publishing schedule has slipped behind a little I propose to get up to date by publishing Issues 3 and 4 together in one bumper issue, nominal cover date June/September.

The Facts of Life

This is the third and fourth free newsletter, and the cost of producing and distributing it now we have so many members is becoming an increasingly large financial burden. So far Greenbank Electronics have helped out with some of the costs (duplication, envelopes, postage and the like) but it is hard to call ourselves an independent group while we are receiving financial support. Also Greenbank have their direct obligations to customers to consider, and must be getting pretty schizophrenic trying to serve both groups of people.

I therefore propose that a subscription fee be charged, to be payable on 30th December each year, although new members will be accepted at any time. It is difficult to budget accurately, (particularly as I don't know how much "shrinkage" in numbers there will be once membership isn't free!), but I propose the following charges:

Payment by Banker's Order	£6.00
Payment by any other means	£7.50

The reason for the substantial discount for payment made by banker's order is simply to encourage members to use this method if they possibly can. The commodity I have in shortest supply is my time, and the banker's order method for subscriptions is the one which I think will take least time to administer. This issue of Interaktion includes a membership renewal form, and banker's order for your use.

On the next page is a proposed constitution - of course this is very flexible and can be modified to suit whatever the majority of users want. Rather than go to the formality of electing a decision-making committee at this stage, I will wait for some "feedback" to see whether or not this is what most people want. One of the great risks with user-group committees in my view is that the committees tend to take over, and the ordinary members begin to feel left out. I think it is better if we're all in it together, as one big happy family, but do telephone or write to me with your views.

Peter Vella

Interaktion User Group - Constitution

1. The aims of the group are to encourage the understanding of the inner workings of computers in general, and specifically the Interak computer(s).

2. A subscription fee is charged, to be paid annually on 30th September, although membership may begin at any time of the year. The fee entitles the member to one year's supply of the "Interaktion" newsletter, (which it is intended to be published quarterly), use of the Group's Library of books, data sheets etc., any software which can be made available, and also a Hardware "library" (tools, test equipment etc.), if one can be established. For all these services a small additional fee may be charged, so that the few will not be enjoying facilities at the expense of the many.

3. The group will attempt to negotiate special arrangements on any items of hardware or software, to make available things which might not otherwise be available to members (e.g. special Interak versions of well-known software products which can be sold under licence).

4. The group may purchase selected items of hardware or software, to form the basis of newsletter articles, for evaluation purposes, or as demonstration pieces to show to other users or potential users.

5. If the users think it appropriate, stands can be taken at exhibitions, to publicise the system and the group.

6. Similarly, if there is a demand, special trips to exhibitions and the like can be arranged. A social secretary can be appointed if members would like to widen their interests beyond simple Interak matters.

7. Local groups of users can be set up, and regular meetings held if desired.

8. A register can be maintained of "Good Samaritans" who can help new users in difficulties - for example to provide local help to save users the cost of a long distance peak rate call to their supplier.

9. Any other ideas, activities: let me know, I am here to help, we can do whatever the majority want.

Peter Vella

ASSEMBLING YOUR THOUGHTS.

When writing programs there are occasions which call for the shortest possible processing time. This is one area where assembly language reigns supreme. Programs run at assembly-language speeds are up to 300 times faster than their BASIC equivalents. How about memory requirements? An Assembly language program held in 4K of memory would require 24K if written in BASIC. These then are the main reasons for spending time learning about and using ASM 32.

All computer systems (including INTERAK) are made up of three rather distinct parts. The CPU, or Central Processing Unit, is the chief controller of the computer system. It fetches and executes instructions, carries out arithmetic calculations, moves data between the other parts of the system, and in general, controls all sequencing and timing of the system. Our CPU is the Z80.

The MEMORY of the system holds a computer's program or programs and various types of data.

The I/O, or input/output devices of the system, allow a user to talk to the computer system in a manner with which he is familiar, such as a typewriter-style keyboard or display of characters on a TV screen.

The main area of concern to us today will be the CPU as this is the part of the system we wish to program using ASM 32. The CPU has a vocabulary which it can interpret and act upon, this vocabulary is known as the instruction set. The Z80 CPU in INTERAK has some 158 different instructions in its instruction set. These can be broken down into the following major groups:-

*Load and Exchange
These instructions move data internally between CPU registers (discussed later) and external memory. All of these instructions must specify a source location from which the data is to be moved and a destination location. The source location is not altered by a load instruction. Exchange instructions can trade the contents of two registers.

*Block Transfer and Search
These allow the transfer of a block of memory of any size to be moved fron any location to any other by the use of a single instruction. They also allow a block of memory of any size to be searched for any 8 bit character. Once the character is found the instruction is automatically terminated to save CPU time.

*Arithmetic and Logical
These operate on data held in the registers or external memory locations. The results of the operations are placed in the A register and flags or markers are set according to the result.

*Rotate and Shift
Allows the changing of bit position within a Byte of data. For example if the Byte 1010 0101 is rotated left it becomes 0100 1011.

*Bit Manipulation
Allows any bit in a register to be tested, set or reset.

*Jump, Call and return
These are used the same way as GOTO, GOSUB and RETURN in BASIC and modify the sequence in which a program is executed.

*Input/Output
As the name suggests these instructions are used to communicate with the Keyboard, Tapes and the like. The Z80 can have up to 256 I/O devices.

*CPU Control Instructions
Allow various options and modes related to interrupts.

As mentioned earlier we are going to be concerned mainly with the CPU and thefore a few lines must now be used to look inside this complex chip. For our programing tasks the CPU can be considered as a group of memory cells each capable of holding one or two bytes of data. These memory cells are known as registers of which there are two main groups.

General Purpose Registers : This group contains six one Byte registers, the A,B,C,D,E,F,H and L registers. Sometimes it is necessary for us to work on double byte data in which case we "pair up" registers into the AF,BC,DE and HL register pairs. Each of these general registers has a shadow called the ALTERNATE register, these are shown in books on Assembly language as A',B',C',D',E',F',H',L'.

Special Purpose Registers: The second group of registers includes the INTERRUPT VECTOR reg (I), MEMORY REFRESH reg (R), two index registers (IX) and (IY), STACK POINTER reg (SP) and the PROGRAM COUNTER reg (PC).

The I and R registers are one byte registers whereas the IX,IY SP and PC are two bytes wide.

The A and F registers are of particular interest and have special functions, the A is in fact the Accumulator used for many of the arithmetic functions, the F is the flag register that is used to indicate the result of certain functions. The flags :- Bit 0 of the F reg is the carry flag and is set when an arithmetic or logical action causes an overflow ie a result greater then 255. Bit 1 is the ADD/SUBTRACT flag used for BCD arithmetic (I shall not go into that) .Bit 2 is the P/V flag and gives the parity of the data. Bit 5 is the half carry flag. Bit 6 is the Zero flag which if the result of a test is true or the result of an arithmetic function is zero. Bit 7 is the Sign flag to tell us if the result of a function is positive or negative.

Having got all that out of the way we can now start on the real work. What I propose is to work through a small program with you, making comments as I think of them. I assume your Interak is on and ready: type in the command R; you should see a pretty box showing the contents of the registers I have mentioned. When we run our program we will use this facility to see how things have gone.

Now load ASM 32 (or ASM 64) then
E 1000 followed by carriage return. (Now for some shorthand):
PROMpt C>

Now type E (clears the work space) PROM C> (get the idea?)

TYpe I (puts us in the insert mode) PROM 1

TY ORG 8000H (this tells ASM that the program we are writing is to run from address 8000H [H for Hex.]) PROM 2

TY LOAD 8000H (ASM will eventually produce some code for us - this tells him the first memory location in which to save it.) PROM 3

TY LD A,2AH (this loads the A REGister with the ASCII code for an "*" we could have used this line LD A,"*" but why make it easy?) PROM 4

TY LD HL,OF80H (guess what? - this loads the HL pair with OF80H which just happens to be a position on the screen.) PROM 5

TY LD (HL),A (Put the contents of the A REG into the address pointed to by HL; so 2AH is placed into address OF080H - i.e. the screen. (Whenever we see (nn) the brackets mean the address pointed to by nn.) PROM 6

TY DB OFFH (this sets a break point at the end of our program see the ZYMON manual. DB stands for Define Byte, so we make the current address = to FFH. Note the leading 0 - this is to tell ASM that the information is a number and not a label [more about labels later]. PROM 7

TY END (tells ASM that this is the last line of our program). PROM 8

TY . (gets us out of the insert mode). PROM C>

We are now ready to let ASM proceed with its work which is to produce machine code.

TY A (asks ASM to start assembly). PROM C>

TY V (this says assemble to video).

Having got all that out of the way we can now start on the real work. What I propose is to work through a small program with you, making comments as I think of them. I assume your Interak is on and ready: type in the command R; you should see a pretty box showing the contents of the registers I have mentioned. When we run our program we will use this facility to see how things have gone.

Now load ASM 32 (or ASM 64) then
E 1ØØØ followed by carriage return. (Now for some shorthand):
PROMpt C>

Now type E (clears the work space) PROM C> (get the idea?)

TYpe I (puts us in the insert mode) PROM 1

TY ORG 8ØØØH (this tells ASM that the program we are writing is to run from address 8ØØØH [H for Hex.]) PROM 2

TY LOAD 8ØØØH (ASM will eventually produce some code for us - this tells him the first memory location in which to save it.) PROM 3

TY LD A,2AH (this loads the A REGister with the ASCII code for an "*" we could have used this line LD A,"*" but why make it easy?) PROM 4

TY LD HL,ØFØ8ØH ~~0F80H~~ (guess what? - this loads the HL pair with ~~0F80H~~ FØ8ØH which just happens to be a position on the screen.) PROM 5

(Assembler needs leading Ø if hex number begins with A-F)

TY LD (HL),A (Put the contents of the A REG into the address pointed to by HL; so 2AH is placed into address ØFØ8ØH - i.e. the screen. (Whenever we see (nn) the brackets mean the address pointed to by nn.) PROM 6

TY DB ØFFH (this sets a break point at the end of our program see the ZYMON manual. DB stands for Define Byte, so we make the current address = to FFH. Note the leading Ø - this is to tell ASM that the information is a number and not a label [more about labels later]. PROM 7

TY END (tells ASM that this is the last line of our program). PROM 8

TY . (gets us out of the insert mode). PROM C>

We are now ready to let ASM proceed with its work which is to produce machine code.

TY A (asks ASM to start assembly). PROM ~~C>~~ OPTION >_

TY V (this says assemble to video).

On the screen you should now see something like this:-

```
1                       ORG ~~LOAD~~ 8ØØØH
2                       LOAD ~~ORG~~ 8ØØØH
3  8ØØØ  3E2A                LD  A,2AH
4  8ØØ2  218ØFØ              LD HL,ØFØ8ØH
5  8ØØ5  ~~7E~~ 77             LD (HL),A
6  8ØØ6  FF                  DB ØFFH
7                      END
C>
```

Column one is the assembly line number.
Column two is the address in memory.
Column three data. (known as OBJECT CODE)
Column four assembler code (Known as SOURCE CODE)

If you have a printer, TY A then E and you will be provided with a hard copy of our program.

TY Z (takes us back to ZYMON).

TY T 8ØØØ (you should see 8ØØØ 3E 2A 21 8Ø FØ ~~7E~~ 77 FF).

TY E 8ØØØ

We have run our program. If all is well you should have the pictures of the registers and an * on your screen. The HL REG should have ~~80F0~~ FØ8Ø in it, A REG 2A and the PC REG 8ØØ6; this is the step at which our program stopped so you can see how it is the PC Program counter that is used by the CPU to keep note of the instruction address to be executed.

TY E 1ØØØ (back to ASM) PROM >C

TY P1Ø (this says print 1Ø lines of program: we don't have 1Ø lines so it prints 7 then EOF End Of File)

Let's make it "more flashy". At the moment if we want to alter the program for say a different screen location it would mean altering the main body of the program. It's quite small at the moment but say it was several 1ØØ lines long (boring) the same with the character. We can in fact replace all reference to numbers with labels - the improved version would read like this:-

```
1          ORG 8ØØØH
2         LOAD 8ØØØH
3
4
5        SCREEN:EQU ~~0F0800H~~ ØFØ8Ø
6        CHR   :EQU 2AH    ;This an *
7
8    ;Main Program
9
```

(listing continues on next page)

On the screen you should now see something like this:-

```
1                          LOAD 8000H
2                          ORG  8000H
3  8000  3E2A              LD   A,2AH
4  8002  2180F0            LD  HL,0F080H
5  8005  7E                LD  (HL),A
6  8006  FF                DB 0FFH
7                   END
C>
```

Column one is the assembly line number.
Column two is the address in memory.
Column three data. (known as OBJECT CODE)
Column four assembler code (Known as SOURCE CODE)

If you have a printer, TY A then E and you will be provided with a hard copy of our program.

TY Z (takes us back to ZYMON).

TY T 8000 (you should see 8000 3E 2A 21 80 F0 7E FF).

TY E 8000

We have run our program. If all is well you should have the pictures of the registers and an * on your screen. The HL REG should have 80F0 in it, A REG 2A and the PC REG 8006; this is the step at which our program stopped so you can see how it is the PC Program counter that is used by the CPU to keep note of the instruction address to be executed.

TY E 1000 (back to ASM) PROM >C

TY P10 (this says print 10 lines of program: we don't have 10 lines so it prints 7 then EOF End Of File)

Let's make it "more flashy". At the moment if we want to alter the program for say a different screen location it would mean altering the main body of the program. It's quite small at the moment but say it was several 100 lines long (boring) the same with the character. We can in fact replace all reference to numbers with labels - the improved version would read like this:-

```
1           ORG 8000H
2          LOAD 8000H
3
4
5          SCREEN:EQU  0F0800H
6          CHR   :EQU  2AH    ;This an *
7
8      ;Main Program
9
```

(listing continues on next page)

```
    10    OUTCHR:LD A,CHR
    11              LD HL,SCREEN
    12              LD (HL),A
    13       DB OFFH;STOP
    14
    15    END
```

Well that's better; lines 3 4 are just to give us space and are produced by pressing RETURN while in the insert mode without any other input. Lines 5 and 6 are used to give values to certain labels, the ; is used as REM is in BASIC and is just a note for us. Line 10 has been given a name and this could be used to CALL, or jump to this line from some other part of the program. Let's edit our existing program to the new version.

```
TY T (takes to the top of the file ) PROM >C

TY D (down one line) PROM C>  2      LOAD 8000H

TY D                 PROM C>  3      LD A,2AH

TY I (Insert mode)        PROM 3

TY RETURN                 PROM 4

TY RETURN                 PROM 5

TY SCREEN:EQU OF080H      PROM 6

TY CHR:EQU 2AH            PROM 7

TY RETURN                 PROM 8

TY ;Main Program          PROM 9

TY RETURN                 PROM 10

TY  .                     PROM C>

TY   N     (replace current line)  PROM 10

TY   OUTCHR:LD A,CHR      PROM C>
```

You can carry on like this till all the amendments are done.

Well that is all for now, next issue we will push further along the trail. If you have any comments let me know.

Peter Vella

Hex. Dump 1, a program called "HEXDUMP II" (Confusing isn't it?).

A program by C.G.Evans which allows you to dump machine code to your printer using the command D. You are asked to give your code a name and this also is printed. Control-P is used (after the name is entered) to turn the printer on and off. (Before using this program modify the end of ZYMON 2 as given in the first line below):

```
07FF  44 00 00 00 09 FF 09 FF A7

1000 00 FILE NAME HEX DUMP
S=0900 E=0AFF

           HEX DUMP           PAGE - 01

0900  21 C6 0A CD DA 09 CD E5 09 21 D1 0A CD DA 09 CD
0910  F6 09 22 F1 0A CD C4 09 21 D4 0A CD DA 09 CD F6
0920  09 22 F3 0A AF 32 F0 0A 06 10 0E 40 CD CA 09 2A
0930  F1 0A ED 5B F3 0A B7 ED 52 D2 66 00 21 D7 0A CD
0940  CA 09 CD CA 09 C5 06 08 CD C4 09 10 FB C1 CD DA
0950  09 C5 06 08 CD C4 09 10 FB C1 21 E8 0A CD DA 09
0960  3A F0 0A 3C 32 F0 0A CD A2 09 CD CA 09 CD CA 09
0970  2A F1 0A CD BB 09 CD C4 09 7E 23 22 F1 0A CD A2

0980  09 CD C4 09 2A F1 0A ED 5B F3 0A B7 ED 52 D2 66
0990  00 2A F1 0A 10 E3 CD CA 09 06 10 0D 20 D2 0E 40
09A0  18 9A F5 CB 3F CB 3F CB 3F CB 3F CD D0 09 CD 4F
09B0  0A F1 E6 0F CD D0 09 CD 4F 0A C9 7C CD A2 09 7D
09C0  CD A2 09 C9 3E 20 CD 4F 0A C9 3E 0D CD 4F 0A C9
09D0  C6 30 FE 3A FA D9 09 C6 07 C9 7E 23 CD 4F 0A 7E
09E0  FE 00 20 F6 C9 21 D7 0A CD 2D 0A 77 FE 0D 28 03
09F0  23 18 F5 AF 77 C9 CD 2D 0A CD 25 0A CB 27 CB 27

0A00  CB 27 CB 27 67 CD 2D 0A CD 25 0A B4 67 CD 2D 0A
0A10  CD 25 0A CB 27 CB 27 CB 27 CB 27 6F CD 2D 0A CD
0A20  25 0A B5 6F C9 D6 30 FE 0A D8 D6 07 C9 C5 D5 E5
0A30  16 00 DB 40 C6 80 30 03 57 18 F7 7A B7 E1 D1 C1
0A40  28 EB FE 10 20 09 3A E7 0A 2F 32 E7 0A 18 DE E5
0A50  F5 FE 0D 20 1C 3A E7 0A B7 20 3F 3E 0D CD BB 0A
0A60  3E 0A CD BB 0A C5 06 03 AF CD BB 0A 10 FA C1 18
0A70  29 D6 1F F2 79 0A F1 E1 C9 C6 1F 2A E5 0A 77 23

0A80  22 E5 0A F5 3A E7 0A B7 20 07 F1 CD BB 0A C3 92
0A90  0A F1 7D FE E0 28 03 F1 E1 C9 21 C0 F2 22 E5 0A
0AA0  C5 D5 21 20 F0 11 00 F0 01 C0 02 ED B0 EB 36 20
0AB0  23 7D E6 1F 20 F8 D1 C1 F1 E1 C9 F5 DB 06 CB 4F
0AC0  28 FA F1 D3 07 C9 46 49 4C 45 20 4E 41 4D 45 20
0AD0  00 53 3D 00 45 3D 00 00 00 00 00 00 00 00 00 00
0AE0  00 00 00 00 00 C0 F2 FF 50 41 47 45 20 2D 20 00
0AF0  00 00 00 00 00 00 00 00 00 00 FF FF FF FF FF FF
```

(end of hex. dump)

Hex. Dump 2: AVALANCHE

This is one for the kids (I won't say what age!), sent in by D.Burns.

Using the < and > characters you have to avoid the falling rocks.

```
A000  DD 21 00 00 21 00 F0 11 01 F0 01 FF 02 36 20 ED
A010  B0 06 01 11 4F 00 D5 C5 21 F0 F2 36 1D E5 ED 5F
A020  C1 D1 A3 D5 C5 3C 87 47 CD F2 04 E1 CD E2 06 FE
A030  2C 20 0D 36 20 2B 7D FE DF 20 01 23 36 1D 18 0F
A040  FE 2E 20 0B 36 20 23 7D FE 00 20 01 2B 36 1D FE
A050  1B CA 77 00 ED 5F CB 47 28 28 ED 5F 1F 3C 1F EE
A060  7F 17 C1 D1 04 4F 78 FE 06 20 0B 14 7A FE 05 20
A070  03 1D 16 00 3E 01 47 81 D5 C5 E6 1F 16 F0 5F 3E

A080  56 12 E5 21 DF F2 11 FF F2 01 DF 02 ED B8 21 00
A090  F0 11 01 F0 01 1F 00 36 20 ED B0 E1 DD 23 7E 36
A0A0  1D FE 56 C2 1D A0 21 00 F0 11 01 F0 01 60 00 36
A0B0  20 ED B0 21 27 A1 11 08 F0 01 08 00 ED B0 11 10
A0C0  27 DD E5 E1 AF ED 52 38 03 3C 18 F9 19 C6 30 32
A0D0  10 F0 11 E8 03 AF ED 52 38 03 3C 18 F9 19 C6 30
A0E0  32 11 F0 11 64 00 AF ED 52 38 03 3C 18 F9 19 C6
A0F0  30 32 12 F0 11 0A 00 AF ED 52 38 03 3C 18 F9 19

A100  C6 30 32 13 F0 7D C6 30 32 14 F0 21 2F A1 11 4B
A110  F0 01 0F 00 ED B0 CD E2 06 28 FB FE 4E CA 77 00
A120  FE 59 20 F2 C3 00 A0 53 43 4F 52 45 20 3D 20 41
A130  4E 4F 54 48 45 52 20 47 41 4D 45 20 3F 20
```

(End of hex. dump.)

Important note: the following article is one of an intended series, discussing the implementation of CP/M on Interak. Although the "FDC-1" card is in preparation, Greenbank tell us that they are not yet in a position to take orders for it, nor the manual mentioned in the article. (The article was written around a prototype version "XFDC-1", laboriously home made by the author R.Eldridge.)

CP/M ON THE INTERAK COMPUTER

PART 1

INTRODUCTION

This article assumes that you have a 48k Interak computer fitted with an XFDC-1 card. You have at least one eight inch disk drive and a standard distribution CP/M diskette. With these items you are now in a position to implement CP/M onto your computer.

At this point you are probably quite worried. You have spent a lot of money and you have yet to see any results for it. Well try to stop worrying, CP/M is a very well thought out system and great fun to use. I suspect that you will be very glad you did it when it's done.

First of all I would like to make some suggestions.

BEFORE YOU BEGIN

1..Ensure that your standard distribution diskette is write protected. There is NEVER going to be a need to write to that disk. To write protect the diskette expose the notch that is to the left of the head run area.

2..Buy a good book on using CP/M. I bought THE CP/M HANDBOOK WITH MP/M written by RODNAY ZAKS and published by SYBEX. I don't say that this is the best, although it may be, it's just the book that I bought and still find very useful.

3..Join the CPMUGUK, this is the CP/M user group for the United Kingdom. It produces a newsletter and can be a source of good ideas etc. It also has a vast library of software, which is supplied at the copying cost only. Pascal and many other goodies are in there. To join this group write in the first instance to: The UK CP/M User Group, 11 Sun Street, Finsbury Square, London EC2M. The editor's name is Andrew Clark and he is always ready to help a newcomer.

4..Buy a box of PRE-FORMATTED eight inch single sided diskettes. When you start you have no way of formatting a disk, and so it is essential that you buy them pre-formatted for single density CP/M. Later on, when you are up and running, the Interak library can provide you with a format utility. (I will just describe formatting a little, in case you don't understand it. If you do then please skip to the next paragraph. Disks that are used by CP/M systems are specific to the drives used. In our case they are eight inch single sided and are IBM-compatible. With this type each disk is divided into tracks and sectors. There are 77 tracks each divided into 26 sectors. The tracks are concentric rings around the disk whilst the sectors are divisions of each track. Each sector has to contain its own pre-written data before CP/M can read or write to it. A formatter will write that information onto a blank diskette.)

5..Sit down and read the manuals that you got with your copy of CP/M. Of great importance, at this stage, is The CP/M 2 Alteration Guide. It is especially worth finding your way around this before you start. Don't try to memorise it. Just read through and try to grasp some of the points in it. A great deal of this information will be difficult to understand, but don't worry it will gradually become clearer as you proceed. I only began to understand a lot of it towards the end of my install. Even as I write this, I am still unravelling some of it.

6..Before you proceed with the CP/M install, carry out the following simple test on your virgin disk hardware. This will familiarise you with its operating concept and at least in part prove that it works.

BASIC TESTS TO THE XFDC-1 CARD
Before any serious work is attempted with your new disk system and in order to ensure that the CP/M construction is successful I propose to run a simple test on the XFDC-1/drive A pair. Please only proceed with the implementation after the following tests have been run.

The access to the disks using the XFDC-1 card is extremely simple and only ZYMON is needed.

As you read this section please have to hand both the Greenbank manual for the XFDC-1 card, and the Data sheet for the Western Digital WD2797-02 Floppy disk chip.

First consider the port allocations for the XFDC-1 card.

PORT	READ	WRITE
80	Status	Command
81	Track	Track
82	Sector	Sector
83	Data	Data
84	Poll	
85	Conr	Conw
86	Opts	

Using the ZYMON monitor we can pass data along these ports and carry out simple disk drive commands.

My test will simply seek to track 40 hex, load the head, verify that track 40 has been found, and restore to track 0.

If and when the following test works you have a good chance that the disk hardware is OK. Although what follows is not an exhaustive test, it does say several things about your the disk subsystem. For example, it proves that the seek and restore commands work. The action of verifying track 40 will have involved a read, and so the read circuits are working. Finally it proves that the controller chip is responding correctly to the port instructions. i.e. the addressing is correct.

THE FIRST DISK TESTS
Insert a PRE-FORMATTED disk into drive A.
Using ZYMON's Port command write the following data to the XFDC ports :-

PORT	DATA	COMMENTS
80	01	Restore
83	40	Data to seek track 40
80	1D	Seek command.

The drive should now seek to track 40 hex.

Now read the data at port 80.
Please reference table 4 of the floppy chip data sheet where you can see status bit allocations for the type 1 command that we have just executed. You can now determine if any faults have occurred and take the required corrective action. An error has occurred if one or more of the following bits are set to a logic 1:

ERROR CONDITION BITS

BIT	ERROR
7	Drive is not ready
4	A seek error has been detected
3	A CRC error was detected
0	The command is unfinished

Bit 2, the track 0 bit, should not be set. If it is then the seek did not take place!

Several valid bits may be set, and combinations of these all mean that the seek was successful.

GOOD SEEK BITS

BIT	MEANING
6	Diskette is write protected
5	The head is loaded
1	Index detected

After you have succeeded in the seek, write to port 80, the data byte 01. The drive will restore to track 0. Check the status by reading port 80 and if all is well proceed with the remainder of this article. If you have a problem then I suggest that you proceed no further until the fault has been rectified and the above tests have run clean.

THE TASK AHEAD

CP/M is supplied for operation on the Intel MDS-800 microcomputer. It is the 20K version that is supplied. As it is it will not work on the Interak computer. Our task is to customise it to run on the Interak computer. This customisation process has been designed in by the builders of CP/M. The major part of the CP/M 2.0 Applications manual describes this process.

Please consider the following information:

CP/M can be divided into four major parts. These are :-

1....A LOADER
2....CCP
3....BDOS
4....BIOS

1...The loader reads CP/M into the computer's store and passes control to it after power switch on. This is called a COLD BOOT.

2...CCP is the Command Control Processor. It is in charge of the computer's operations. It never deals directly with the system hardware. Whenever an input/output task is needed it always works through the BDOS or the BIOS.

3...BDOS is the Basic Disk Operating System. It processes all disk I/O for the CCP. BDOS also is unable to deal with the system hardware, it passes all such tasks to the BIOS.

4...BIOS is the Basic Input Output System. It is hardware dependent and interfaces the CCP BDOS pair to the particular computer system in which they reside.

We are allowed to apply our own BIOS and LOADER to replace those in the standard CP/M. If we do this correctly then CP/M will work on our own computer. The people who invented CP/M intended us to make this alteration. The system has been constructed to ease this customisation. The copyright of CP/M will not be infringed by appling these changes and the parts created by the customiser belongs to, and may be sold by him. What you must not do is supply a complete CP/M system, without licence, to some other person.

Our task is simply to provide CP/M with a customised LOADER and BIOS. Put this down to disk and off we go.

The LOADER goes at the front of CP/M and the BIOS goes at the back. It has all been designed to make this customisation process quite simple, and providing you go slowly it will work perfectly. Even I managed to do it!

If you wish you may follow the procedure as described in the CP/M 2.0 Applications manual. But I found it simpler to go about it in a slightly different way. This article describes the way I implemented CP/M, and does not necessarily follow the method described by the authors of the CP/M 2.0 Applications manual.

LET US BEGIN
We shall first of all build a CP/M to run in 20K of store. Later we shall convert this to a 48K system. We have to construct the 20K system first because that's what's on your standard distribution disk.

If you have read the CP/M 2 alteration guide then you will understand that the construction of a first system can be described in 5 simple steps as :-

1..Read the first two tracks from the standard distribution diskette into memory.

2..Replace, in the in store copy of CP/M, the Intel MDS-800 LOADER with the Interak LOADER.

3..Replace, in the in store copy of CP/M, the Intel MDS-800 BIOS with the Interak 20K BIOS.

4..Write the new system from store to the first two tracks of ANOTHER diskette.

5..Provide some way of automatically loading CP/M at computer power on time. (COLD BOOT)

STEP 1
We need to read tracks 0 and 1 into the machine memory.

Call this task GETSYS.

Let us now consider the task in some detail :-

1..The 20K CP/M will be read to :-

```
        HEX. ADDR     CP/M COMPONENT
         3380........LOADER
         3400........CCP
         3C00........BDOS
         4A00........BIOS
         4D7F........end
```

2..Our GETSYS will read the first two tracks from the diskette to 3380 hex. and return control to ZYMON.

3..ZYMON lives at 0000-07FF and the 20K CP/M is going to be placed at 3380-4D7F. Then let the GETSYS reside at 8000. (All numbers in hex.)

4..It is worth ensuring that GETSYS will work before we go live. So I propose that we also write a PUTSYS. This will allow us to write something to a BLANK diskette, read it back, and check that it's OK. So let GETSYS begin at 8000 hex and PUTSYS begin at 8003 hex. These locations will jump to the real routines.

THE GETSYS/PUTSYS ROUTINES
These can either be entered using ASM or you can punch the code directly using ZYMON. When it's in save it to a clean tape so that it is always available.

```
Remember E 8000.....enters GETSYS....Read CP/M
         E 8003.....enters PUTSYS....Write CP/M
```

Before we use these routines on the CP/M diskette we should test them. That I will describe after the program code.

```
            ;TITLE GETSYS/PUTSYS

             ORG 8000H      ;Getsys Putsys code

            ;Equates

0080         DSTAT:  EQU 80H        ;Disk status
0080         DCOMM:  EQU 80H        ;Disk command
0081         DTRAC:  EQU 81H        ;Disk track register
0082         DSECT:  EQU 82H        ;Disk sector register
0083         DDATA:  EQU 83H        ;Disk data register
0084         DPOLL:  EQU 84H        ;Disk polling
0085         DCONF:  EQU 85H        ;Disk configuration
0088         READS:  EQU 88H        ;Read sector command
00AC         WRITS:  EQU 0ACH       ;Write sector command
0059         STEPIN: EQU 59H        ;Step in command
000D         RESTOR: EQU 0DH        ;Restore command
00D0         RESET:  EQU 0D0H       ;Reset command
0066         ZYMON:  EQU 66H        ;ZYMON's warm start

            ;Code begins

8000 C30680 GETSYS: JP READ         ;Branch to read 0 & 1
8003 C35880 PUTSYS: JP WRITE        ;Branch to write 0 & 1

            ;::::::::::::::::::::::::::::::::::::::::::::::
            ; GETSYS CODE.  DISK A, READ TRACKS 0 & 1     :
            ;::::::::::::::::::::::::::::::::::::::::::::::

            ;Read CP/M from tracks 0 & 1 of drive A

8006 310090 READ:    LD SP,9000H    ;Initialise the stack

            ;First configure for an 8 inch, SSSD drive.

8009 3E0C            LD A,0CH       ;Set drive A as 8" SSSD
800B D385            OUT (DCONF),A  ;Configure drive to read

            ;Execute a floppy disk controller reset.
            ;Wait for 28 micro secs after a reset command.

800D 3ED0            LD A,RESET     ;Load reset command
800F D380            OUT (DCOMM),A  ;Reset disk controller
8011 0600            LD B,0         ;Loop count = 256
8013 10FE    RRES:   DJNZ RRES      ;Wait for reset to end

            ;Set HL to point to destination of the disk data.
            ;Set the C register to the disk polling port.

8015 218033          LD HL,3380H    ;Load point for disk data
8018 0E84            LD C,DPOLL     ;Set disk polling register
```

```
                 ;Command a restore to track zero on drive A

801A 3E0D                 LD A,RESTOR    ;To restore drive A
801C D380                 OUT (DCOMM),A  ;Restore drive A
801E ED78        RWAITO:  IN A,(C)       ;Get disk poll data
8020 E601                 AND 1          ;Is command complete?
8022 28FA                 JR Z,RWAITO    ;Loop if not complete

                 ;First sector to read is 1

8024 1E01        RSTART:  LD E,1         ;Sector to read

                 ;Read the sector in E to (HL). 128 bytes.

8026 7B          RDSEC:   LD A,E         ;Get sector to read
8027 D382                 OUT (DSECT),A  ;Pass this to the FDC
8029 3E88                 LD A,READS     ;Get read sector command
802B D380                 OUT (DCOMM),A  ;Command a read sector
802D ED50        RDRQ:    IN D,(C)       ;Get poll status
802F 28FC                 JR Z,RDRQ      ;Loop till data ready
8031 DB83                 IN A,(DDATA)   ;Get disk data
8033 77                   LD (HL),A      ;Store disk data
8034 23                   INC HL         ;Point to next address
8035 FA2D80               JP M,RDRQ      ;Loop till sector all in

                 ;Get disk status byte.  If an error was sensed
                 ;then loop and try the entire read process again.
                 ;The routine will lock into this loop if the
                 ;drive has a permanent error due to a disk fault
                 ;condition.

8038 2B                   DEC HL         ;Adjust store pointer
8039 DB80                 IN A,(DSTAT)   ;Get ending status
803B E69D                 AND 9DH        ;Mask for error codes
803D C20680               JP NZ,READ     ;If error, goto start!

                 ;The JP NZ,READ instruction is the path taken on
                 ;an error condition.  If you have problems with
                 ;your hardware then you may change this branch to
                 ;go to your own routine.  Normally with well set
                 ;up disks and a correctly aligned XFDC-1 card, no
                 ;errors should be noticed.
                 ;So RECHECK your setup!

                 ;HERE IF. The sector was read in with no error.
                 ;If more sectors are left on this track go
                 ;and read another sector.

8040 1C          RSOK:    INC E          ;Staticise next sector
8041 7B                   LD A,E         ;Get next sector
8042 FE1B                 CP 27          ;At end of track?
8044 38E0                 JR C,RDSEC     ;If no read next sector
```

```
                ;The entire track has been read.  If that was
                ;track 0 then step to track 1.  Else if that
                ;was track 1 exit, task complete.

8046 DB81                 IN A,(DTRAC)  ;Get the current track
8048 3D                   DEC A         ;Was it 1?
8049 CA6600               JP Z,ZYMON    ;If yes exit done

                ;Seek to track 1

804C 3E59                 LD A,STEPIN   ;Get stepin command
804E D380                 OUT (DCOMM),A ;Step to track 1
8050 ED78       RWAIT1:   IN A,(C)      ;Get seek status
8052 E601                 AND 1         ;Mask for command done
8054 28FA                 JR Z,RWAIT1   ;Loop till seek complete
8056 18CC                 JR RSTART     ;Go and read track 1

                ::::::::::::::::::::::::::::::::::::::::::::::::
                ; PUTSYS CODE.  DISK A, WRITE TRACKS 0 & 1    :
                ::::::::::::::::::::::::::::::::::::::::::::::::

                ;Write CP/M to tracks 0 & 1 of drive A

8058 310090     WRITE:    LD SP,9000H   ;Initialise the stack

                ;First configure for an 8 inch, SSSD drive.

805B 3E0C                 LD A,0CH      ;Drive : A : 8" : SSSD
805D D385                 OUT (DCONF),A ;Configure A : 8" : SSSD

                ;Execute a floppy disk controller reset.
                ;Then wait for 28 microsecs after reset command.

805F 3ED0                 LD A,RESET    ;Load controller reset
8061 D380                 OUT (DCOMM),A ;Command controller reset
8063 0600                 LD B,0        ;Delay 28 micro secs for
8065 10FE       WRES:     DJNZ WRES     ;controller to reset

                ;Set HL to point to the data source.
                ;Set the C register to the disk polling port.

8067 218033               LD HL,3380H   ;Source point for CP/M
806A 0E84                 LD C,DPOLL    ;Disk polling register

                ;Command a restore to track zero on drive A

806C 3E0D                 LD A,RESTOR   ;Load disk restore command
806E D380                 OUT (DCOMM),A ;Command a restore to 0
8070 ED78       WWAIT0:   IN A,(C)      ;Loop till seek complete.
8072 E601                 AND 1
8074 28FA                 JR Z,WWAIT0
```

```
                  ;First sector to write is 1

8076 1E01    WSTART: LD E,1            ;E reg has current sector

                  ;Write 128 bytes from HL to the disk sector in E

8078 7B      WSEC:   LD A,E            ;Get the current sector
8079 D382            OUT (DSECT),A     ;Pass this to the FDC
807B 3EAC            LD A,WRITS        ;Get write sector command
807D D380            OUT (DCOMM),A     ;Command Write sector
807F 7E              LD A,(HL)         ;Get first data byte
8080 23              INC HL            ;Point to the next
8081 ED50    WDRQ:   IN D,(C)          ;Check ready to write?
8083 28FC            JR Z,WDRQ         ;Loop if not ready
8085 D383            OUT (DDATA),A     ;Write data to FDC
8087 7E              LD A,(HL)         ;Get next data byte
8088 23              INC HL            ;Point to the next
8089 FA8180          JP M,WDRQ         ;Loop if not done.

                  ;Sector has been written.
                  ;Get disk status byte.  If an error was sensed
                  ;then loop and try the entire write process
                  ;again.  The routine will lock into this loop
                  ;if the drive has a permanent error due to
                  ;a disk fault condition.

808C 2B              DEC HL            ;Adjust store pointer
808D 2B              DEC HL            ;To point to next data
808E DB80            IN A,(DSTAT)      ;Get disk status byte
8090 E69D            AND 9DH           ;Mask for write errors
8092 C25880          JP NZ,WRITE       ;If error, retry the write

                  ;The JP NZ,WRITE instruction is the path taken on
                  ;an error condition. It will retry from the start
                  ;forever!! So if you have problems with your
                  ;hardware then you may change this branch to
                  ;go to your own routine. Normally with well set
                  ;up disks, and a correctly aligned XFDC-1 card, no
                  ;errors should be noticed.
                  ;So RECHECK your setup!

                  ;HERE IF. The sector was written with no error.
                  ;If more sectors are left on this track go
                  ;and read another sector.

8095 1C      WSOK:   INC E             ;Advance the sector
8096 7B              LD A,E            ;Load next sector
8097 FE1B            CP 27             ;All sectors done?
8099 38DD            JR C,WSEC         ;If no go for next sector
```

```
                   ;The entire track has been written. If that was
                   ;track 0 then step to track 1. Else if that
                   ;was track 1 exit task complete.

809B DB81                    IN A,(DTRAC)  ;Load current track
809D 3D                      DEC A         ;Was it track 1?
809E CA6600                  JP Z,ZYMON    ;If yes then exit done.

                   ;Seek to track 1

80A1 3E59                    LD A,STEPIN   ;Load step in command
80A3 D380                    OUT (DCOMM),A ;Command Step in.
80A5 ED78          WWAIT1:   IN A,(C)      ;Wait for command to
80A7 E601                    AND 1         ;terminate.
80A9 28FA                    JR Z,WWAIT1   ;Wait loop
80AB 18C9                    JR WSTART     ;Go for next track write.

80AD                         END



DCOMM     0080  DCONF     0085  DDATA     0083  DPOLL     0084
DSECT     0082  DSTAT     0080  DTRAC     0081  GETSYS    8000
                PUTSYS    8003  RDRQ      802D  RDSEC     8026
READ      8006  READS     0088  RESET     00D0  RESTOR    000D
RRES      8013  RSOK      8040  RSTART    8024  RWAITO    801E
RWAIT1    8050  STEPIN    0059  WDRQ      8081  WRES      8065
WRITE     8058  WRITS     00AC  WSEC      8078  WSOK      8095
WSTART    8076  WWAITO    8070  WWAIT1    80A5  ZYMON     0066

ERRORS=0000
```

ENTER GETSYS AND PUTSYS
To prepare the two routines type them into the computer's main memory using ZYMON's Modify command. In order to be certain that no typing errors have been made I suggest that you then type the two routines again but this time put them starting at address 9000 hex. After you have done that you can use ZYMON's verify routine to check one against the other. Type V 8000 80AD 9000. ZYMON will now list any differences and you can correct any errors re-verifying until the two blocks of code compare. This procedure will give you the maximum chance of having no typing errors as it's unlikely that you will make the same error twice.

Now that the code block 8000 80AD is in store it would be well worth saving it to tape. S 8000 80AD. The next thing to do is to reload the tape with L 9000. This is simply to ensure that the save was a good one. Having done that label the tape 8000 GETSYS 8003 PUTSYS and place it to one side.

TEST GETSYS AND PUTSYS
We have a part tested disk system and we wish to build our operating system with it. To me it is now essential to validate the disk hardware. The GETSYS/PUTSYS pair are an ideal tool for this task and I propose that at this point we carry out a series of tests.

DISK SUBSYSTEM TEST
TEST 1..Put the CP/M diskette to one side.
TEST 2..Place a PRE FORMATTED blank diskette into drive A.
TEST 3..Use ZYMON to F 3300 4FFF FF
TEST 4..Use ZYMON to E 8003.
This will execute PUTSYS and write 3380-4DFF to tracks 0 and 1. It will take no more than 10 seconds to return to ZYMON's warm start point. If this happens then the write was good and you may go to test 5.

FAULT AT TEST 4..A fault is indicated if ZYMON has not warm started after 60 seconds. In this case there is a permanent write error being sensed and I suggest that you press the front panel Reset switch.

FAULT 4 ACTION 1..Use ZYMON to F 3380 4DFF 00.
This presets the memory area to 00. It was all FF.

FAULT 4 ACTION 2..Use ZYMON to E 8000.
This will read tracks 0 and 1 to store at 3380-4DFF. It will take less than 5 seconds to return to ZYMON's warm start point. If ZYMON's warm start point is not entered within 60 seconds press the front panel reset switch.

FAULT 4 ACTION 3..Use ZYMON to tabulate the store 3300 4FFF. As you know we tried to write FF's to the disk and after clearing the area to 00 we have tried to read it back. By examining the area you should be able to determine if the write action was partially successful or not. You must then try to deduce the hardware problem and take action to correct it. I am unable to help here because so many things can be at fault. All I can suggest is that you check the XFDC-1 card set up. Then use ZYMON to load the FDC's internal registers for a seek to track 40, with track verify on. Use ZYMON to command the seek and then to get the status byte back. This will perhaps give you a clue to the fault. After rectification return to Test 1 and re-do the sequence.

TEST 5..Use ZYMON to F 3000 4FFF 00.
The write was good so we now clear the store area prior to a read back test.

TEST 6..Use ZYMON to E 8000.
This executes GETSYS which should return to ZYMON's warm start point within 5 seconds. If this happens then the read was good and you may proceed to test 7.

FAULT 6 ACTION 1..If ZYMON's warm start point has not been entered after 60 seconds, press the front panel reset switch.

FAULT 6 ACTION 2..Use ZYMON to tabulate the store 3300 4FFF. As you know we have written FF's to the disk and after clearing the area to 00 we have tried to read them back. By examining the area you should be able to determine if the read action was partially successful or not. You must then try to deduce the hardware problem and take action to correct it. I am unable to help here because so many things can be at fault. All I can suggest is that you check the XFDC-1 card set up. Then use ZYMON to load the FDC's internal registers for a seek to track 40, with track verify on. Use ZYMON to command the seek and then to get the status byte back. This will perhaps give you a clue to the fault. After rectification return to Test 1 and re-do the sequence.

```
TEST 7..Use ZYMON to  F 3300 4FFF FF
        Use ZYMON to  F 3380 33FF 55
        Use ZYMON to  F 3400 347F AA
        USE ZYMON to  C 3380 347F 3480.
```
This has formed a 1k data block consisting of 256*55:256*AA:256*55:256*AA.

TEST 8..Use ZYMON to E 8003.
This will write the data to the disk.

```
TEST 9..Use ZYMON to  F 3300 4FFF 00
        Use ZYMON to  E 8000.
```
This will read the data back.

TEST 10.Use ZYMON to check that the data read back is as was written. If so all is well and you may proceed. If not then the fault should be rectified and the entire sequence redone from TEST 1.

I will assume that the above tests have run OK for the remainder of the CP/M implementation process.

READ IN THE CP/M DISKETTE

The disk system has been validated. The GETSYS and PUTSYS routines are now trusted. It is time to read our standard distribution diskette.

Place the standard distribution diskette into drive A.

Use ZYMON to F 3300 4FFF FF

Use ZYMON to E 8000.
This will read CP/M into store.

Remove the standard distribution diskette from drive A and place it to one side.

Place a PRE-FORMATTED diskette into drive A.

Use ZYMON to E 8003.
This will write CP/M to the disk.

Remove this disk from drive A. Label this disk as follows:-
COPY 20K CP/M STANDARD DISTRIBUTION.

Place this disk to one side. It will be used to reload CP/M and so protect the master disk.

Place another PRE-FORMATTED diskette into drive A.

Use ZYMON to E 8003.
This will again write CP/M to disk.

Use ZYMON to C 3380 4D7F 2380.
This will place a copy of CP/M to store at 2380 hex.

Use ZYMON to E 8000.
This will read back the previously written CP/M from the disk.

Use ZYMON to V 3380 4D7F 2380.
This will finally confirm the validity of the mechanisms, both hard and soft, to date.

Remove the diskette from drive A and label it as follows:-
INTERAK 20K CP/M. WORK DISK 1.
FIRST CUSTOMISATION. date.

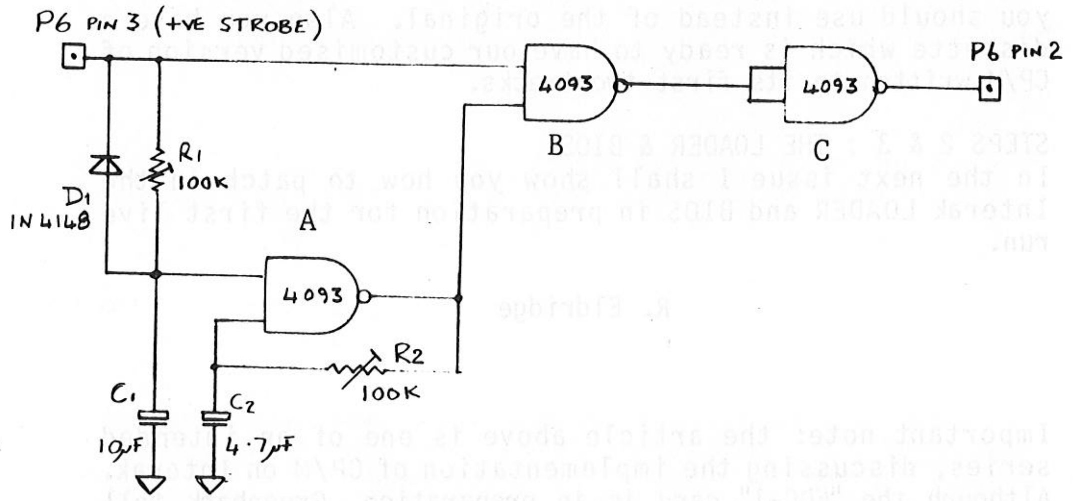
STEP 1 COMPLETED
You now have a copy of the standard distribution CP/M. This you should use instead of the original. Also you have a diskette which is ready to have our customised version of CP/M written to its first two tracks.

STEPS 2 & 3 : THE LOADER & BIOS
In the next issue I shall show you how to patch in the Interak LOADER and BIOS in preparation for the first live run.

R. Eldridge

Important note: the article above is one of an intended series, discussing the implementation of CP/M on Interak. Although the "FDC-1" card is in preparation, Greenbank tell us that they are not yet in a position to take orders for it, nor the manual mentioned in the article. (The article was written around a prototype version "XFDC-1", laboriously home made by the author R.Eldridge.)

AUTO KEYBOARD REPEAT FOR THE LKP-1



P6/3 is a positive strobe and when a key is pressed goes high.

The B gate output goes low and produces a high at C gate output. If the key is released P6/3 goes low and the output follows.

If the key is held down C1 is charged via R1 to enable the oscillator comprising gate A, C2 & R2. Gate B is enabled and the strobe output oscillates until the key is released. D1 gives rapid discharge of C1 to enable rapid keystroking. R1 & R2 may be fixed by trial and error and the whole circuit may be fitted into the patch area on the PCB. Gate A must be a Schmitt and the spares in U3 & U10 might be used but R1 & R2 will then need to be less than about 2k and result in a large C values.

R1 sets delay time before repeat; R2 sets repeat frequency.

The facility works on all keys and is particularly useful for drawing lines, cursor movements and scanning memory using the T mm command to scan right through memory to find where memory has got to etc.

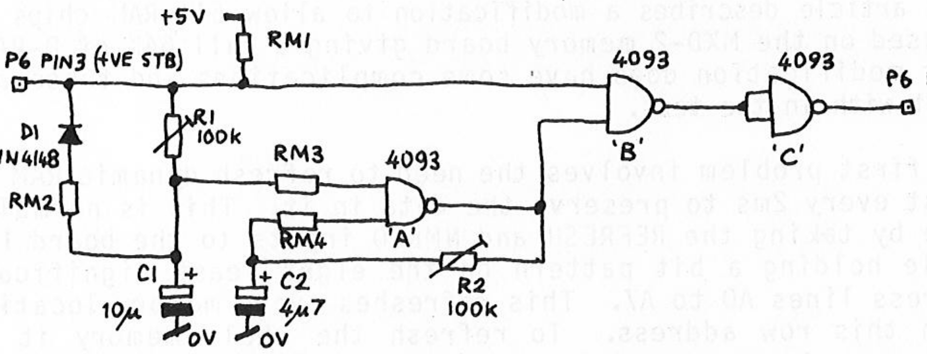
The spare gate inputs should be grounded for CMOS devices.

Contributed by: Don Horn

(If anyone local to me please get in touch 42 The Newlands, Wallington, Surrey, phone 01 669 2892.)

(For some comments from Greenbank see next page)

Comments on the Auto Keyboard Repeat for the LKP-1



I have added 4 resistors labelled RM1-4 to the circuit and will use these to identify some places where I would like to make a few comments. In general note that the modification as described by Don Horn only applies to keyboards which present a positive edge on their strobe output, and maintain the strobe output high all the time the key is held down. Not all keyboards are so obliging!

RM1
If P6/3 is driven from LS TTL without a pull-up resistor, the voltage for a "1" could be as low as 2.7V, The input voltage needed for a worst-case 4093 has to be greater than 4 volts or so. The pull-up would increase the 2.7V pretty well to the full 5V. Of course the pull-up is in fact built in to the LKP-1 design, but not if the strobe is inverted (i.e. P6 pin 1 connection - see LKP-1 manual circuit diagram).

RM2
If the strobe line is driven by an ordinary LS TTL output, when the strobe goes low C1 is shortcircuited via the diode D1 and the driving TTL output. Ordinary LS TTL short circuit current is around 20-100mA, but the data sheet say that the duration of the short circuit must not exceed 1 second. RM2 would limit the current to a few milliamps only and thus make life easier for all components.

RM3,4
If C1 and/or C2 happen to be charged when the power is removed they will have to discharge through the input protection diodes built into the 4093. My Motorola data sheet says the current through any pin of the 4093 must not exceed 10 mA, so a value of 470 ohms for RM3,4 would guarantee this. (Of course things aren't actually as bad as I make out because the +5V power rail goes down gradually so the C1 and C2 discharge currents may not be too great anyway). I make such heavy weather of all this because I have always been very interested in the design of circuits for long term reliability - excess currents cause the migration of metal ions gradually to places where they do no good, and failure can occur years later.

David Parkins, Greenbank Electronics

Modifying the MXD-2 Memory Board to use 4164 64K RAM chips

This article describes a modification to allow 64K RAM chips to be used on the MXD-2 memory board giving a full 64K of D-RAM. This modification does have some complications and these are dealt with in the text.

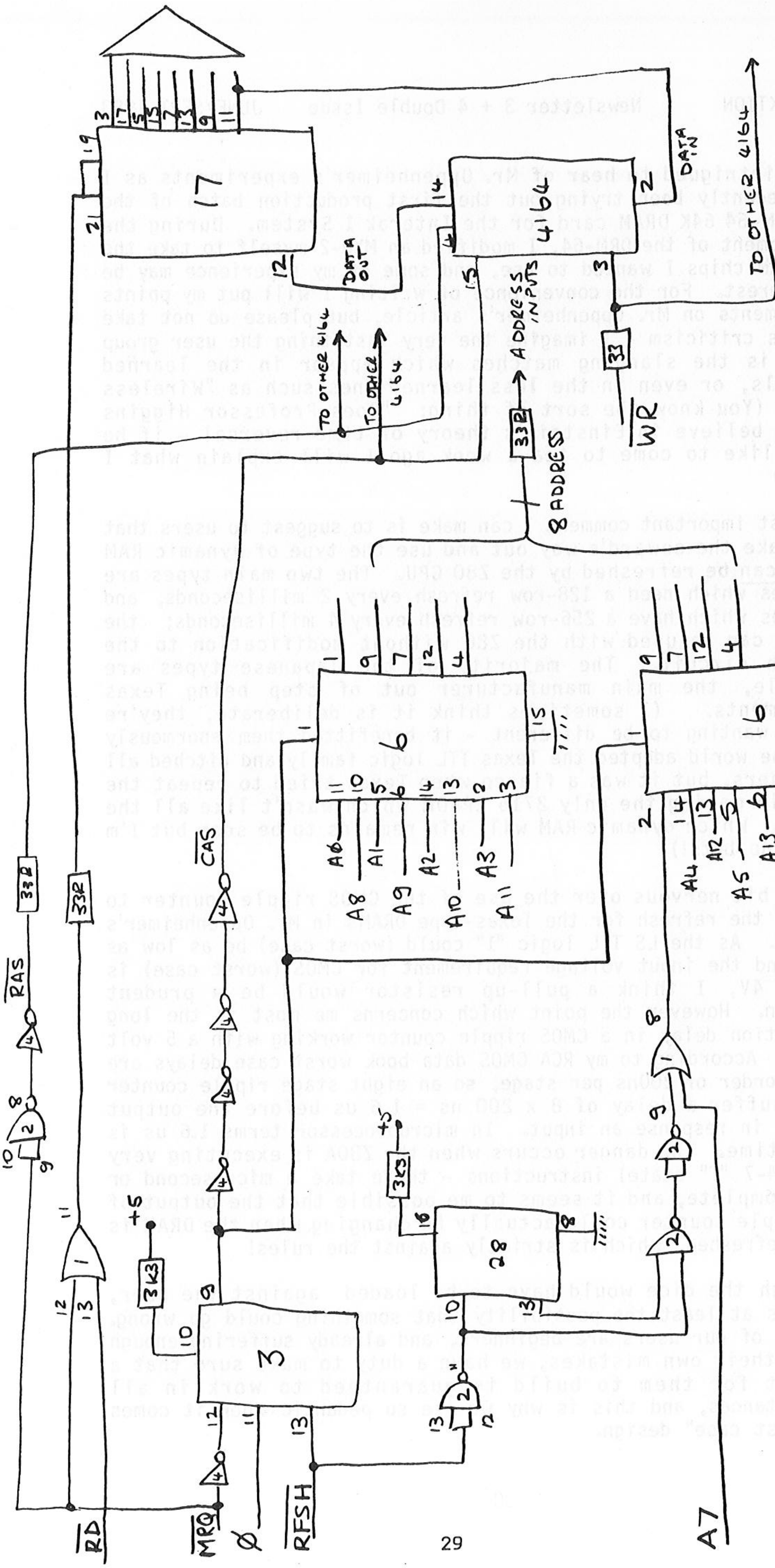
The first problem involves the need to refresh dynamic RAM at least every 2ms to preserve the data in it. This is normally done by taking the REFRESH and NMREQ inputs to the board low while holding a bit pattern on the eight least significant address lines A0 to A7. This refreshes every memory location with this row address. To refresh the whole memory it is necessary to repeat this operation with every possible row address, that is, with each of the 256 possible 8-bit patterns being placed on lines A0 to A7 in turn. The Z80 performs this function automatically whilst the CPU is excecuting an instruction internally, the row addresses being generated by an internal counter (R-register). Unfortunately the R-register ony cycles from 0 to 127, that is seven bits while to refresh 64K eight-bit patterns cycling from 0 through to 255 are needed. To generate theis eighth bit this project uses the eighth output of a binary ripple counter to supplement the CPU's own refresh signals.

As the memory is no longer divided into 16K blocks, the 2 to 4 line decoder IC5 is no longer needed as every memory access will be applicable to this board. It should be possible to mount the binary ripple counter in the space previously occupied by IC5. The ripple counter is in fact a CMOS device, chosen partly due to the shortage of TTL ripple counters and partly because one happened to be available. It does however perform quite satisfactorily and the system has been running 24 hours a day for several months.

To avoid conflicts on the bus when reading from memory locations within the video RAM the NRDS input was disconnected from the VDU-K board. Clearly the contents of the D-RAM and the video RAM should be identical when the system is running, but it was thought that conflicts could occur at switch on. This modification is otherwise totally transparent to the computer when the system is running because reading a screen address provides the data out of the D-RAM which is identical to that held in the corresponding video RAM location.

Richard Oppenheimer

(Editor's Note. The circuit diagram is on the next page, redrawn from the sketch provided by Richard; on the page after that are the comments from Greenbank Electronics as they have been selling the MXD-2 card for many years although they did not design it.)


RD
MREQ
Ø
RFSH
A7
RAS
CAS
+5
3k3
33Ω
33Ω
33Ω
33
7
3
6
8
28
4164
WR
DATA OUT
DATA IN
TO OTHER 4164
TO OTHER 4164
TO OTHER 4164
ADDRESS INPUTS
8 ADDRESS
A0
A1
A2
A3
A4
A5
A6
A8
A9
A10
A11
A12
A13
A14
A15

I was intrigued to hear of Mr. Oppenheimer's experiments as I have recently been trying out the first production batch of the new DRM-64 64K DRAM card for the Interak 1 System. During the development of the DRM-64, I modified an MXD-2 myself to take the 64K DRAM chips I wanted to use, and some of my experience may be of interest. For the convenience of writing I will put my points as comments on Mr. Oppenheimer's article, but please do not take them as criticism - I imagine the very last thing the user group wants is the slanging matches which appear in the learned journals, or even in the less learned ones such as "Wireless World" (You know the sort of thing: "Does Professor Higgins really believe in Einstein's theory of time reversal - if he would like to come to tea a week ago I will explain what I mean"!)

The most important comment I can make is to suggest to users that they take the coward's way out and use the type of Dynamic RAM which <u>can</u> be refreshed by the Z80 CPU. The two main types are the ones which need a 128-row refresh every 2 milliseconds, and the ones which have a 256-row refresh every 4 milliseconds; the former can be used with the Z80 without modification to the refresh circuit. The majority of the Japanese types are suitable, the main manufacturer out of step being Texas Instruments. (I sometimes think it is deliberate, they're always wanting to be different - it benefitted them enormously when the world adopted the Texas TTL logic family and ditched all the others, but it was a fiasco when Texas tried to repeat the trick in making the only 2716 EPROM which wasn't like all the others. Which dynamic RAM will win remains to be seen but I'm taking no bets!)

I am a bit nervous over the use of the CMOS ripple counter to provide the refresh for the Texas-type DRAMs in Mr. Oppenheimer's circuit. As the LS TTL logic "1" could (worst case) be as low as 2.7V, and the input voltage requirement for CMOS (worst case) is around 4V, I think a pull-up resistor would be a prudent addition. However the point which concerns me most is the long propagation delay in a CMOS ripple counter working with a 5 volt supply. According to my RCA CMOS data book worst case delays are in the order of 200ns per stage, so an eight stage ripple counter could suffer a delay of 8 x 200 ns = 1.6 us before the output changed in response an input. In microprocessor terms 1.6 us is a long time. The danger occurs when the Z80A is executing very short (4-7 "T" state) instructions - these take a microsecond or so to complete, and it seems to me possible that the output of the ripple counter could actually be changing when the DRAM is being refreshed, which is strictly against the rules!

Although the dice would have to be loaded against the user, there is at least the possibility that something could go wrong. As many of our users are beginners, and already suffering enough due to their own mistakes, we have a duty to make sure that a circuit for them to build is guaranteed to work in all circumstances, and this is why we are so pedantic when it comes to "worst case" design.

(The dice have been loaded against our competitors as well: I often enjoy a malicious chuckle at a very famous computer which was first advertised as being able to work with "all" television receivers, but later had to claim instead to be suitable only for "most" TVs - I know what I'd think If I'd bought one!)

It is worth mentioning that our calculations show that RAMs with an access time of 150ns have to be used in our 64K design. Although in practice slower RAMs have given no trouble to us it would be a braver man than I who would say take a chance with the price of 64K RAMs being what it is!

There is one inheritance of the MXD-2 which has been carried forward into the 64K modification, and that is the inadequacy (in theory) of the "RAS Precharge Time". At 4 MHz the Z80 provides less than 100ns; most 64K RAMs require more. The book "1980 Memory Data Book and Designers' Guide" (no longer in print, but in stock at Greenbank) provides more information on this topic. The design of the new DRM-64 includes a method of extending the precharge time, so that it can be right in theory as well as in practice.

The idea of avoiding bus contention at all times is one which I endorse wholeheartedly, so I am glad not to have to debate this topic.

In my experiments with a modified MXD-2, (and indeed the early prototypes for the DRM-64) I found the track layout and dynamic RAM drive circuits to be inadequate, causing odd corruptions of data unless the RAM card was positioned right next to the CPU card, at one end of the rack. I would be most interested to learn if Mr. Oppenheimer suffered any of this difficulty, and what his cure was if he did. (It is to ease this problem that users will find one side of the DRM-64 extensively endowed with supply-line planes, which helps quite a bit in reducing the "noise" which aggravates the corruption of data in the RAMs.)

I believe that Mr. Oppenheimer also has a modification in mind which allows the use of a new line to be called "RAMDIS", which would let the 64K RAM card overlay any items such as EPROMs which are in the way. As this would need alteration to the bus, the solution adopted on the DRM-64 card for Interak 1 is simply to provide DIL switches which can be used to make 4K "holes" in the memory map where conflicts would otherwise have arisen. The next phase of development of Interak will permit many different items to share the memory space, and re-arrangement will take place when required by memory-mapping software. Rather than generating and using a new RAMDIS signal unwanted items will be forced off the bus simply by denying them their existing NMREQ signal. (Although it is only in dreamland at present a new CPU card design is taking shape which will permit access to a memory space of some 16,384K, and yet remain compatible with humble Interak 1 with its mere 64K.)

David M. Parkins, Greenbank Electronics

THE INTELGRAPH - AN INTELLIGENT GRAPHICS VDU

The Intelgraph is a single board video display unit giving a 512 by 256 bit mapped display that does not occupy any processor memory space. Access to the unit is via one of four interface connections, ranging from 20mA loop to direct port mapping. The 131,072 programable pixels allow 80 by 25 text and true high resolution grapics to be freely mixed. Lines, arcs and circles can be drawn by direct command to the internal processor and single pixel control is fully programmable, allowing commands such as set, reset, invert, point, and several others to be utilised in the creation of high density video artwork.

An advertisement describing the unit can be found in the February 1983 edition of Personal Computer World page 296. Amongst other things this shows that the device can be bought in various stages of input option with one of two screen formats. Namely 512 by 256 and 256 by 256. This review is based on the 512 by 256 format.

The device is sold under the name INTELGRAPH and can be obtained from COMPUTER AIDS, 33 Bearsden Crescent, Hinkley, Leicester, LE10 0SQ.

On the face of the above description it seems that the Intelgraph could be a worthwhile enhancement to the Interak computer, and so I set about a trial implementation on my 48k twin disk system. What follows is a description of my experiences with the Intelgraph.

THE CARD

My prototype arrived ready assembled and so I can't comment on how difficult it is to build. Suffice it to say that all of the component positions are marked on the board, which is fully through hole plated and should be very easy to assemble. My one complaint is that the chips seem to be upside down to the markings, but once that is realised it represents no real problem. The manual seems good, and full layout and circuit details are given. This is a definate plus point as it allows you to understand the whys and hows of the hardware, also if you have a faulty component having the circuit means you have a chance to locate it and get the unit working. I personally dislike the idea of not including circuits with hardware and the Intelgraph designers can be congratulated on producing such clear and readable logics.

THE INTERFACE

The Intelgraph offers several interface options. These are serial 20mA loop, serial RS232C, serial TTL, and parallel. Each interface can be set up via on board DIL switches and the interface wiring is hooked up on DIP headers to on-board sockets provided for that purpose.

The power interface is via a sturdy terminal block; the Intelgraph requires 1A at 5v for parallel operation with + and - 12v being needed if the serial options are implemented.

The VDU interface is either from an on-board modulator for connection to a normal TV set or by direct feed to a video monitor.

The three serial interface options can be selected to operate at baud rates ranging from 50 to 19200 baud, with DIP switched start, stop and parity selection.

The parallel interface can be accessed either by ports or through the memory map and requires two locations. One of these is used as a read only status port and the other is a write only data port. The data port is used to send print information, escape sequences and command data to the Intelgraph.

The status port gives the host computer information pertaining to the condition of the video module. Bits 7,6, and 5 are avaliable in all of the various modes with bits 4,3,2,1, and 0 being used in the parallel mode only. These last five bits are known as the message bits, and carry error message codes. The status bits are listed below :-

7	READY. When high, ready to accept data/commands
6	ERROR. When set an error message is in bits 4-0
5	BUFFER. Set when the buffer is empty.
4-0	MESSAGE. The messages bits are assigned :-

4 3 2 1 0	
0 0 0 0 1	Illegal command
0 0 0 1 0	X parameter out of range
0 0 0 1 1	Y parameter out of range
0 0 1 0 0	Not used
0 0 1 0 1	Escape sequence error
0 0 1 1 0	Error in an arc command

The remaining codes are not used.

CONNECTING UP

I decided to use the parallel interface and on the next page I have drawn the circuit that was used to connect the INTELGRAPH to my computer. This circuit connects E0 hex as the status port, and E1 hex as the data port.

My interface was built using the circuit below. FIG 1

```
                                             5v ---+----
                                                   !
                                                 ----
                                                 !  !
  INTERAK COMPUTER                               !1k!
  (system bus)                                   !  !
                                 ------------------
                                 ! 74LS136 !
  A7 ---------------- 1!                   !3 ----+
        0v ---------- 2!                   !      !
  A6 ---------------- 13!                  !11 ---+
        0v --------- 12!     IC1           !      !
  A5 ---------------- 4!                   !6 ----+
        0v ---------- 5!                   !      !
  A4 ---------------- 10!                  !8 ----+
        5v ---------- 9!                   !      !
               5v-- 14!                    !7 -0v !
                                 -----------      !
                                 -----------------!
                                 ! 74LS136 !      !
  A4 ---------------- 13!                  !11 ---+
        5v --------- 12!                   !      !
  A3 ---------------- 1!                   !3 ----+
        5v ---------- 2!     IC2           !      ! select (H)
  IORQ -------------- 4!                   !6 ---------------+
        5v ---------- 5!                   !                 !
        5v ---------- 9!                   !10 --------------+
    +---------------- 8!                   !
    !          5v-- 14!                    !7--0v
    !                            -----------
    !                            -----------
    ! select (L)                 ! 74LS139 !   port E0
    +---------------- 1!                   !4 ---------+
                                 !             port E1 !
  A0 ---------------- 2!                   !5 -----+   !
                                 !   IC3           !   !
  A1 ---------------- 3!                   !6 ---NC  !  ! INTELGRAPH
                                 !                 !  !  ! interface.
                                 !                 !7 ---NC  !  !
               5v-- 14!                    !8 --0v  ! +---STATUS
                                 -----------        +------ DATA
  5v --------------------------------------------------------- 5v
  0v --------------------------------------------------------- 0v
  WR ------------------------------------------------------ RD/WR
  D7 --------------------------------------------------------- D7
  D6 --------------------------------------------------------- D6
  D5 --------------------------------------------------------- D5
  D4 --------------------------------------------------------- D4
  D3 --------------------------------------------------------- D3
  D2 --------------------------------------------------------- D2
  D1 --------------------------------------------------------- D1
  D0 --------------------------------------------------------- D0
```

POWER UP AND SIGN ON

At power up the sceen clears and a cursor is seen at the top left hand corner, it is gently flashing. This is a programmable option and can be changed by the user to Not flashing, flashing or invisible. It is also possible to implement a reset line (another port perhaps) which will cause this power up sequence to be under program control. It should be noted that to cause the reset to happen you need to drop pin 15 of socket 3 to a low level. This is slightly more difficult to do than is usual because pin 15 has a 47 micro farad capacitor to ground and a 1k resistor to +5v, connected to it. These are to create the power on reset, and your logic signal must be capable of dropping this point to a low level.

THE TEST SOFTWARE

On my INTERAK I use CP/M, and so I created a CP/M driver routine to handle the INTELGRAPH. I include the driver for your interest.

```
;CP/M to INTELGRAPH driver.
;Conout,(sub),sends the C reg to the VDU.
CONOUT: CALL VDUCHK         ;Check vdu free
        LD A,C              ;Get character to print
        OUT (OE1H),A        ;Pass it to the vdu
        RET                 ;Exit done
VDUCHK: IN A,(OEOH)         ;Get status
        BIT 7,A             ;Ready
        JR Z,VDUCHK         ;No - loop till ready
        BIT 6,A             ;Error ?
        JR NZ,VERROR        ;Yes - do error handler
        RET                 ;Exit ready to print data
VERROR: LD A,'?'            ;On error print question mark
        OUT (OE1H),A        ;Print the error marker.
        JR VDUCHK           ;Try to continue
```

The Intelgraph will directly print ASCII codes to the next sequential screen position, plus it handles all of the normal control codes, and so this routine will work quite well for the text only CP/M BIOS function. The only problems I experienced, with the above, was loss of characters, but when I reduced the processor speed to 2 MHz this did not happen, to find out why I examined the cases of lost characters and determined that only under continuous string transfers were characters missing. The subroutine that was concerned is shown below.

```
;Prnmsg,(sub),prints the string (HL) till (HL) = 0
PRNMSG:  LD A,(HL)        ;Read string character
         OR A             ;Set up flags on char
         RET Z            ;Return if zero char
         LD C,A           ;Character to C
         CALL CONOUT      ;Print character in C
         INC HL           ;Advance the pointer
         JR PRNMSG        ;Branch for next char
```

This routine, above, rapidly sends text characters pointed to by the HL register pair to the VDU via the CONOUT routine. In the above state it was just too fast for the INTELGRAPH to handle. The INTELGRAPH ignored the odd character. The solution was to add an inter character delay which completely cured the problem. The replacement printstring routine is shown below with its associated delay subroutine.

```
;Prnmsg,(sub),prints the string (HL) till (HL) = 0
PRNMSG: LD A,(HL)         ;Read string character
        OR A              ;Set up flags on char
        RET Z             ;Return if zero char
        LD C,A            ;Character to C
        CALL CONOUT       ;Print character in C
        CALL DELAY        ;Pause
        INC HL            ;Advance the pointer
        JR PRNMSG         ;Branch for next char

DELAY:  PUSH AF
        POP AF
        PUSH AF
        POP AF
        RET
```

There is very little overhead caused by using the delay between characters, and the delay, or one like it, should be called whenever a byte of data is passed to the INTELGRAPH. This includes graphic byte sequences. The only exception is BASIC, as most BASICS are slow enough between byte transfers not to need the delay. So the general rule is; If passing bytes to the INTELGRAPH, by machine code subroutines whilst using a 4MHZ clock, call the delay after each byte. Or use a byte passing subroutine i.e.

```
OUTBYTE: DATA TO VDU       ;Pass the byte in C to the VDU
         CALL DELAY        ;VDU recovery time.
         RET
```

The symptoms of over fast transfers are lots of random printing usually followed by screen lockup, requiring a power off/on to clear.

I also found that the INTELGRAPH required a time, at system power on, to reset and be ready to work. I had to execute the delay routine 512 times, after power on, before I could access the screen processor. This is not a problem as it is only the power on reset circuit timing out.

THE TEXT DISPLAY

By coding a special routine I displayed the ASCII character set on the screen. This is a fairly simple task as the INTELGRAPH accepts the ASCII set between 20 and 7E hex and prints them to the next cursor position. I then examined the construction of the characters with interest (as I have had some experience with the task of character building). The close inspection showed that generally all the characters were well formed, except for four of the lower case characters t,d,g and i. These are slightly out of balance with the remainder of the set. This is unfortunate but it seems only an oversight as, for instance, the q is perfect whereas the g is not. These two sharing the same characteristics suggest that only the selection of dots is in error and not some horrendous fault as yet unrealised. I must also add that my card is a prototype and I suspect that the real released version will have these four characters corrected.

USER DEFINED CHARACTERS

As well as the ASCII set there are 25 user-definable characters that are printed by sending 80 through 98 hex. The term user-definable means that each of these characters may be defined dot by dot, and then printed in their entirety by simply sending the appropriate code to the INTELGRAPH. The first ten of these characters are initialised by the device on power up, but can be redefined if required. This can be very handy as the basic character set can be expanded on a program customisation basis to add the symbol pi for instance.

SUPERSCRIPT - SUBSCRIPT

The two codes A0 hex, and A1 hex, if sent to the INTELGRAPH, cause the next character sent to be superscripted or subscripted respectively, allowing the chemical description of water H_2O or the "raise to the power" function to be correctly displayed. The use of the bottom line of the screen does not affect this function as the bit mapping allows the character so subscripted to remain within the line area: Suppose you wished to print the chemical formula for water, then you would send to the INTELGRAPH the byte sequence 48 A1 32 50, all in hex. The four bytes sent would command :-

48	print ASCII H
A1	subscript the next character
32	print (subscripted) ASCII 2
50	print ASCII O

Notice that only the character immediately following the script command byte is acted upon. Characters following on after the subject byte will be printed in the normal way until another script command is issued.

The inclusion of the scripted character capability, coupled with graphic representation, makes this a very useful device for the physical sciences. You can now plot and describe scientific functions using terminology that is visually correct. Also, many printers now have script and graphic capabilities, allowing screen to hardcopy to become much more representative and valuable.

WORDSTAR COMPATABILITY
Wordstar is a CP/M word processor and so I thought it reasonable to try out the INTELGRAPH with this most demanding of text orientated programs. The following are the changes required to install Wordstar for the INTELGRAPH. This assumes the CONOUT driver previously defined is in your BIOS.

All numbers are in hex. Alter "FROM" values to be "TO" values after replying N to "ARE ALL MODIFICATIONS COMPLETE". See the Wordstar manual for more details on installation.

ADDRESS	FROM	TO	
0248	18	19	25 lines per screen
024C	3D	5B	CLEAD1 second character
0253	00	01	CLEAD2 number of characters
0254	00	3B	CLEAD2 first character
0258	00	01	CTRAIL number of characters
0259	00	63	CTRAIL post line and column char
025E	20	00	LINEOFF line offset
025F	20	00	COLOFF column offset
0260	00	02	ASCUR send 2-digit ASCII numbers
0284	00	01	IVON length
0285	00	0E	IVON highlight on
028B	00	01	IVOFF length
028C	00	0F	IVOFF highlight off
02AE	0A	14	DELCUS delay after cursor set
02AF	05	14	DELMIS delay after other functions
02B5	00	FF	CRBLIV blink on inverse
02D2	40	10	DEL4 post sign on delay

Having completed the install, I brought up the new version of Wordstar and was pleased to find it and the INTELGRAPH very compatible.

Working with the two components, in fact this was written with them, I can at last feel that I have a computer system that is a close approximation to my intentions when I started out. The extra screen space of 80 characters allows much better layouts and lots of elbow room. The performance of the INTELGRAPH during reverse scroll and CTRL C,R activity is quite reasonable and probably compares quite well to serial terminals. Obviously it is slower than my previous VDU 2K, but the extra 16K of RAM that the INTELGRAPH will allow me to fit will more than compensate, as Wordstar will be faster with the extra store.

The point about the extra store is worth a further word or two. On the current INTERAK we are restricted to 48K as a maximum memory size. This is primarily due to the VDU being a memory mapped device, overlaying page F. By implementing the INTELGRAPH the top 16K of store can be freed for RAM, allowing CP/M to be the 64K version. This will enable the homebrew INTERAK to match other Z80 systems, if not excell them.

GRAPHIC DISPLAYS

The INTELGRAPH has 131,072 programmable pixels arranged in a matrix of 512 by 256. These pixels can be lit or unlit. In this way drawings can be constructed of very high density. Also as an enhancement of the above the INTELGRAPH has an internal set of algorithms, driven by a 6502B processor, which allow lines, arcs and circles to be drawn with a few command bytes. This is of considerable use as large parts of any drawing usually contain basic components of curves and lines, so that it is only the odd irregular elements that will require single pixel drawing.

The graphic commands can be divided into two types :-

Linear commands..... 4 bytes per command.
Angular commands.... 7 bytes per command.

LINEAR COMMANDS

Syntax:- Command X coordinate Y coordinate

Commands	Data transfered
Set pixel at X,Y...........................	1C XL XH Y
Clear pixel at X,Y.........................	1D XL XH Y
Test pixel at X,Y..........................	18 XL XH Y
Invert pixel at X,Y........................	15 XL XH Y
Draw from current positon to X,Y...........	1E XL XH Y
Undraw from current position to X,Y........	19 XL XH Y
Invert from current position to X,Y........	17 XL XH Y
Move current position to X,Y...............	1F XL XH Y

Where the first byte is the command in hex.
XL is the X value low.
XH is the X value high.
Y is the Y value.

Note that X is in the horizontal plane, and Y is in the vertical plane. Y has a range of 0 - 255 decimal.

The bytes XH and XL together make a 16 bit integer value whose range is 0 - 511 decimal.

The first four commands, SET, CLEAR, TEST and INVERT, all act on individual pixels and allow the maximum choice of light/dark areas on the display.

The next three commands are DRAW, UNDRAW and INVERT, and these act upon a straight line that exists between the current position of an imaginary graphics pointer and the X, Y values in the command.

The last command in this group is MOVE and this is used to position an imaginary pointer. This pointer represents the starting position of any draw type command and a line drawn will always originate at this pointer position.

A further option exists in that, if a flag is enabled then the X,Y values in the command become relative to the imaginary pointer, such that X and Y can now be expressed as values of positive and negative with respect to that pointer. This means that if the imaginary pointer is set to the screen centre, for example, then you can draw to the positive side (right or up) and to the negative side (left or down) of it. Negative values in this case are two's complement values of X and Y, i.e. -5 is FB.

The relative mode flag is set and reset by use of escape sequences which are discussed later.

ANGULAR COMMANDS
Syntax type A:-
   Command start-angle end-angle radius aspect

Syntax type B:-
   Command reference-angle-low reference-angle-high

```
Draw arc   (syntax A).... A2 STAL STAH EAL EAH RAD APR
Undraw arc (syntax A).... A3 STAL STAH EAL EAH RAD APR
Origin arc (syntax B).... A5 RAL RAH
```

The first byte is the command in hex.
STAH is the start angle high, STAL is the start angle low.
EAH is the end angle high, EAL is the end angle low.
RAD is the arc of radius.
APR is the aspect ratio.
RAH is reference angle high, RAL is reference angle low.
In the above STAH and STAL comprise a 16 bit start angle.
EAH and EAL comprise a 16 bit end angle.
RAH and RAL comprise a 16 bit reference angle.

The first two commands, DRAW ARC and UNDRAW ARC, will act on an arc begining at the specified start angle, using the specified radius and ending at the specified end angle, these angles all take the vertical axis as zero degrees.

The value ASP is the aspect or perspective of the arc. A circular curve is produced with an aspect of 16 whilst an aspect of 0 will produce a straight line, with the intermediate values producing ellipses.

The final command in this group is ORIGIN ARC. This rotates an arc in the X plane such that the zero angle position is rotated anti-clockwise about the vertical axis. For example, suppose an arc is specified as starting at 0 degrees and ending at 30 degrees. With a reference angle of 0 degrees, it will be drawn starting directly above the imaginary reference point set up by the linear move command. If now the origin arc command is used and specifies 45 degrees then the arc will be produced from the 45 degree position about the reference point and extending for the 30 degrees specified in the draw command i.e. to 75 degrees.

I spent considerable time drawing complex curves and lines using the graphic command sequences. They all seem to work as indicated in the user manual, and are very easy to manipulate. However there is one oddity which seems to affect the RAL value in the originate arc sequence, this is when a value of 90 degrees is used to specify a horizontally drawm arc. The result is the same as when 0 degrees are used and had a most odd effect on my drawing. The fault only showed up at 90 degrees, whereas 89 and 91 degrees worked perfectly.

I was slightly displeased with the quality of arc produced by the above commands. Whilst they are substantially correct, there are deviations from the path of circumference, which spoil the true image of the curve commanded. These deviations, although slight, could have been avoided as sufficient pixels do exist along the desired arc of radius. I suspect, but I don't know for sure, that the algorithms used are implemented with integer arithmetic and do not take into acount the fractional components of sine and cosine, so giving rise to a slight tendancy to triangulate.

Rather than use miles of paper to describe the above graphic commands, and because I cannot draw you any examples of the weird and wonderful effects that can be produced, I have included an example program that draws the planet Saturn with its rings. This will give you some idea of the programming requirements of the graphics.

The program includes my mechanisms for handling commands and outputting graphic data. You can see that I have set up a command buffer to hold a three command graphic sequence, consisting of MOVE, followed by ORIGINATE, followed by DRAW ARC. The individual contents of that buffer are now modified by the program and drawn by a single subroutine DRARC, via CONOUT.

This program can easily be altered to draw other things. Just change the front end whilst leaving the subroutines alone. Your front end will set up the desired command sequence, and the existing subroutines do the drawing.

```
                ;TITLE      SATURN
1000 310030  BEGIN: LD SP,3000H ;Initialise the stack
1003 0E0C           LD C,12     ;Command to clear screen
1005 CD7A10         CALL CONOUT ;Clear the screen

                ;Fill the arc table with initial values.
1008 215B10         LD HL,PTAB  ;Arc initial data source
100B 114D10         LD DE,ORGARC ;Arc table pointer
100E 010E00         LD BC,14    ;Length of data
1011 EDB0           LDIR        ;Fill arc parameter table

               ;Construct the planet mass.
1013 065A    PLANET:LD B,90     ;Planet loop counter
1015 3A5210  PL1:   LD A,(RAL)  ;Get reference angle
1018 C602           ADD A,2     ;Advance reference angle
101A 325210         LD (RAL),A  ;Save new reference angle
101D CD6910         CALL DRARC  ;Draw the arc defined
1020 10F3           DJNZ PL1    ;Repeat for 90 arcs

                ;Draw the planet rim.
1022 3E10           LD A,16       ;Aspect for planet rim
1024 325A10         LD (ASP),A    ;Save the rim aspect
1027 CD6910         CALL DRARC    ;Draw the planet's rim

                ;Draw the rings of Saturn.
102A 3E5B    RINGS: LD A,91     ;Ring reference angle
102C 325210         LD (RAL),A  ;Save this as RAL
102F 3EAA           LD A,170    ;Saturn's rings max radius
1031 325910         LD (RAD),A  ;Save this as RAD
1034 3E02           LD A,2      ;Ring aspect value
1036 325A10         LD (ASP),A  ;Save this as ASP
1039 CD6910         CALL DRARC  ;Draw the outer ring
103C 0606           LD B,6      ;Six more rings to draw
103E 3A5910  RINGX: LD A,(RAD)    ;Get current ring radius
1041 D60A           SUB 10        ;Point to next inner ring
1043 325910         LD (RAD),A  ;Save this as RAD
1046 CD6910         CALL DRARC  ;Draw the next ring
1049 10F3           DJNZ RINGX  ;Draw the remaining rings
104B 18FE    FREEZE:JR FREEZE   ;Loop until user resets

               ;Parameter table for DRARC subroutine.
104D 00      ORGARC:DEFB 0        ;Move coordinates command
104E 00      LOWX:  DEFB 0        ;Low X coordinate
104F 00      HIGX:  DEFB 0        ;High X coordinate
1050 00      LOWY:  DEFB 0        ;Low Y coordinate
1051 00      ARCORG:DEFB 0        ;Origin arc command
1052 00      RAL:   DEFB 0        ;Elipse reference low
1053 00      RAH:   DEFB 0        ;Elipse reference high
1054 00      ARCBGN:DEFB 0        ;Draw an arc command
1055 00      STAL:  DEFB 0        ;Start angle low
1056 00      STAH:  DEFB 0        ;Start angle high
1057 00      EAL:   DEFB 0        ;End angle low
1058 00      EAH:   DEFB 0        ;End angle high
1059 00      RAD:   DEFB 0        ;Radius
105A 00      ASP:   DEFB 0        ;Aspect ratio
```

```
105B 1FFF008 PTAB:  DEFB 31,255,0,128  ;Parameter table
105F A50000A        DEFB 165,0,0,162   ;initialisation
1063 0000680        DEFB 0,0,104,1     ;data
1067 3C00           DEFB 60,0
               ;Table ends, subroutines follow.

              ;Drarc, (sub), draw arc from ORGARC table
1069 C5       DRARC: PUSH BC           ;Save BC
106A E5              PUSH HL           ;Save HL
106B 214D10          LD HL,ORGARC      ;Point to parameter table
106E 060E            LD B,14           ;Length of parameter table
1070 4E       DRARC1:LD C,(HL)         ;Get the next parameter
1071 CD7A10          CALL CONOUT       ;Pass it to the Intelgraph
1074 23              INC HL            ;Advance to next parameter
1075 10F9            DJNZ DRARC1       ;Loop pass all parameters
1077 E1              POP HL            ;Restore HL
1078 C1              POP BC            ;Restore BC
1079 C9              RET               ;Exit Arc is drawn

             ;Conout,(sub),sends C reg to the VDU.
107A F5       CONOUT:PUSH AF           ;Save AF
107B CD8610          CALL VDUCHK       ;Check vdu free
107E 79              LD A,C            ;Get character to print
107F D3E1            OUT (OE1H),A      ;Pass it to the vdu
1081 CD9710          CALL DELAY        ;Allow vdu to cycle
1084 F1              POP AF            ;Restore AF
1085 C9              RET               ;Exit
1086 DBEO     VDUCHK:IN A,(OEOH)       ;Get status
1088 CB7F            BIT 7,A           ;Ready ?
108A 28FA            JR Z,VDUCHK       ;No - loop till ready
108C CB77            BIT 6,A           ;Error ?
108E 2001            JR NZ,VERROR      ;Yes - do error handler
1090 C9              RET               ;Exit ready to print data
1091 3E3F     VERROR:LD A,'?'          ;On error print character
1093 D3E1            OUT (OE1H),A      ;Print the error character
1095 18EF            JR VDUCHK         ;Try to continue

             ;Delay, (sub), Intelgraph cycle time
1097 F5       DELAY: PUSH AF           ;11 T states
1098 F1              POP AF            ;21 T states
1099 F5              PUSH AF           ;32 T states
109A F1              POP AF            ;42 T states
109B C9              RET               ;52 T states
                     END
```

SATURN SYMBOL TABLE

```
ASP:    105A   EAL:    1057   PL1:   1015   STAH:  1056
ARCBGN:1054    EAH:    1058   PTAB:  105B   VDUCHK:1086
ARCORG:1051    FREEZE:104B    RINGS:102A    VERROR:1091
BEGIN: 1000    HIGX:   104F   RINGX:103E
CONOUT:107A    LOWX:   104E   RAL:   1052
DELAY: 1097    LOWY:   1050   RAH:   1053
DRARC: 1069    ORGARC:104D    RAD:   1059
DRARC1:1070    PLANET:1013    STAL:  1055
```

SCREEN HANDLING
Under this heading I shall indicate all of the commands that act to configure and control output on the screen. I shall further subdivide this group of commands into two types. These are single byte control commands and escape sequences.

SINGLE BYTE CONTROL COMMANDS	HEX	
Home the cursor	01	
Cursor left	08	(Backspace)
Cursor right	09	
Cursor down	0A	(Line feed)
Cursor up	1A	
Clear screen, Home cursor	0C	
Carriage return, line feed	0D	
Reverse video on	0E	(Until cancelled)
Reverse video off	0F	(Until cancelled)
Delete, shuffle line left	7F	
Superscript	A0	(Mentioned earlier)
Subscript	A1	(Mentioned earlier)

ESCAPE SEQUENCES	SEND THESE FOUR ASCII CHARACTERS
Non blinking cursor	ESC [0 b
Blinking cursor	ESC [1 b
Underlining off	ESC [0 u
Underlining on	ESC [1 u
Character size * 1	ESC [1 w
Character size * 2	ESC [2 w
Character size * 3	ESC [3 w
Character size * 4	ESC [4 w
Character size * 5	ESC [5 w
No auto line feed	ESC [0 l
Auto line feed	ESC [1 l
Draw absolute to pointer	ESC [0 a
Draw relative to pointer	ESC [1 a
Move cursor to row RR, column CC....	ESC [RR ; CC c
Scrolling window, row RR, row RR ...	ESC [RR ; RR r
Define user character DDD	ESC [DDD d

In the last three escape sequences, RR represents a number range of 1 through 24, CC represents a number range of 1 through 80 and DDD represents a number range of 128 through 152. You must send the ASCII codes that make up the number.

For example :-
To move the cursor to row 3 col 22 send the command :-

```
In hex  1B  5B 31 3B 32 32 63
ASCII    ESC [  3  ;  2  2  c
```

To set the scrolling window to row 8 - 22 send :-

```
In hex  1B  5B 38 3B 32 32 72
ASCII    ESC [  8  ;  2  2 r
```

To set the character size to * 5,(* 1 is normal), send :-

```
In hex  1B  5B 35 77
ASCII   ESC [  5  w
```

IN CONCLUSION
If you would like an 80 by 24 VDU with the added bonus of graphics then this unit will provide it. The few problems I have found are all easily correctable and do not detract significantly from the overall quality of the product.

I must warn you to beware of having my first reaction when you connect up the unit. I did not like it. The letters were very small, and the access was unfamiliar. Actually the letters are correctly sized for the display. It was me, having got used to the large VDU K letters, was not used to normal sized characters. Silly I know, but just give the device a little time and it will grow on you. I suspect that in the end you will enjoy using it. The exercise of thinking in pictures and relating them to code so as to produce those pictures, is both demanding and stimulating, and the reward for success is immediate.

If this device becomes widely used on Interak we shall have to consider defining its implementation. Whether it should be serial or port accessed for instance. Also it would need a definite protocol, and that should be as commercially compatible as possible. On the software aspect, implementation should be quite possible, as I believe the new BASIC that Pete Vella is working on will be quite easy to adapt, CP/M is no problem and ASM will convert very simply. So the only thing we need is a monitor program to handle it at the machine code level. I don't think ZYMON can be converted without a major refit.

To sum up my opinion in one sentence is difficult, but if pressed I would say. "Good fun, good value and well worth having."

Bob.Eldridge

More from Your ZYBASIC 2.

ArcSine. Gives:

ArcSine(S), angle whose sine is S, where 0<=S<<1 (uses X,Y internally):

```
3400  X=S: IF ABS ( S)<= 0.707107 G.3450
3410  X=1-S*S:IFX<0 P. S;"IS OUT OF RANGE":S.
3420  W=X/2:Z=0  3430  Y=(X/W-W)/2 : IF (Y=0)+
      (Y=Z) X=W : G. 3450  3440  W=W+Y:Z=Y :G.
      3430
3450  Y=X+X*X*X/6+X*X*X*X*X*0.075+X*X*X*X*X*X*
      X*4.464286E-2
3460  W=Y+X*X*X*X*X*X*X*X*X*3.038194E-2
3470  IF ABS(S)> 0.707107 W=1.570796-W
3480  Y=W*57.29578:R.
```

When run this routine will give ArcSine (S) in degrees as Y and in radians as W.

ArcCosine.

Gives ArcCos(S), angle whose cosine is S, where 0<=S<=1 (uses X,Z internally) and uses the ArcSine subroutine:

```
3500 GOS.3400:Y=90-Y:W=1.570796-W:R.
```

When run gives ArcCos (S) in degrees as Y and in radians as W.

ArcTangent.

Gives ArcTan(S), angle whose tangent is X.

(Uses B,T internally and the value of X is changed); uses the Sign subroutine on the next page.

```
3600  GOS.3700:X=ABS(X):C=0
3610  IF X>1 C=1:X=1/X
3620  A=X*X
3630  B=((2.86623E-3*A-1.61657E-2)*A+4.29096E-2)*A
3640  B=((((B-7.5289E-2)*A+0.106563)*A-0.142089)
      *A+0.199936)*A
3650  A=((B-0.333332)*A+1)*X
3660  IF C=1 A=1.570796-A
3670  A=T*A:C=A*57.29578:R.
```

When run gives ArcTan (S) in degrees as C and in radians as A.

More from Your ZYBASIC 2. (continued)

Sign. (Used with the previous routines).

Gives SGN(X), the sign-component of X.

```
3700 IF X<0 T=-1
3710 IF X=0 T=0
3720 IF X>0 T=1
3730 R.
```

When run T is set to -1 if X is negative, T set to 0 if X = 0 and T set to 1 if X is positive.

P.V

GRAPHICAL PLOTTER.

This program sent in by M. Cottrell for use with ZYBASIC 2 allows the plotting of graphical functions.Instructions are written in to the program and are fairly obvious to use. Briefly, a blank graph is displayed on which can be included:

1. X & Y axis information.

2. Plots of whatever function is inserted into line 170, (presently Y=SINE (X)*A).

3. Individual plots to be linked up by the computer. The plots of functions can be overlaid on top of the linked plots - the linked plots put first. Although the number of pixels is limited it will display a SINE Wave using data V1=20, X=0 , X1=10 , A=20 , B C + D = 0 . The graph can be redrawn without disturbing AXIS information already on display.

```
1 CLS:LINE1
2 P."GRAPH PLOTTING INSTRUCTIONS": P."...........................................":
  P."40 VERTICAL  PLOTS AVAILABLE":
  P."56 HORIZONTAL PLOTS AVAILABLE":P."INPUT DATA V1(0 TO 40)
  SHIFTS ":P."THE HORIZONTAL AXIS UPWARDS.":."SCALING IS
  ACCORDING TO INPUT ":P."VARIABLES.
3 P."X IS INDEPENDANT VARIABLE":P."STARTING VALUE.":
  P."X1 IS INCREMENTAL CHANGE OF X.":
  P."TO ENTER YOUR GRAPHIC FUNCTION":
  P."KEY IN 1 THEN 170 Y=(formula) ":
  P."USING VARIABLES X,A,B,C,D":P."AND RERUN PROGRAM.
4 P."PLOTS MAY ALSO BE ENTERED WITH":
  P."NO FORMULA WHICH WILL BE DRAWN":P."BY THE COMPUTER.":
  P."SELECT FOR DATA,GRAPH,AXIS -":
  P."SCALE OR THESE INSTRUCTIONS BY":P."KEYING IN C...."
7 INK.A:IF  A=#43 GOTO 10
```

GRAPHICAL PLOTTER. (continued)

```
8 IF A=#31 GOTO 280
9 IF A=0 GOTO 7
10 DOFF:CLS:LINE1:GOTO 15
12 P."                            ":
   FOR A1 =4 TO 60:FOR A2 =6 TO 47:RESET(A1,A2):NEXT A2:NEXT A1
15 POKE#F002,#13:POKE#F01E,#12:POKE#F282,#11:POKE#F29E,#10
38 A=0
40 J=#F003:POKE(J+A),#19:A=A+2:IF A<=26GOTO 40
48 A=0
50 J=#F283:POKE(A+J),#19:A=A+2:IF A<=26 GOTO 50
52 A=0
55 J=#F004:POKE (A+J),#17:A=A+2:IF A<=25 GOTO 55
57 A=0
60 J=#F284:POKE(A+J),#16:A=A+2:IF A<=25 GOTO 60
68 A=0
70 J=#F022:POKE(A+J),#18:A=A+28:POKE(A+J),#18:A=A+36
72 IF A<600GOTO70
73 A=0
80 J=#F042:POKE(A+J),#15:A=A+64:IF A<550 GOTO 80
82 A=0
90 J=#F05E:POKE(A+J),#14:A=A+64:IF A<550 GOTO 90
92 A=0
95 FOR A=1 TO 26:J=#F002:J=J+A:IF PEEK(J)=#17 GOTO 100
97 GOTO 105
100 FOR B=1 TO 19: POKE(J+32*B),#18:NEXT B
105 NEXT A
107 J=#F002
110 J=J+64:IF J>#F269 GOTO 180
115 A=0
120 A=A+1:POKE(A+J),#19:A=A+1:IF A>27 GOTO  110
122 POKE(A+J),#1A:GOTO 120
123 P."                            ":LINE 23:GOTO 180
126 P."                            ":J=#F2A2:
    FOR A=1TO29:POKE(J),04
127 INK.X:IF X=0 GOTO 127
128 POKE(J),X:J=J+1:NEXT A
129 J=#F280
131 GOSUB 140
132 IF J<=#F002 GOTO 165
133 J=J-66:GOTO 131
140 FOR A=1TO2:POKE(J),04
142 INK.Y:IF Y=0 GOTO 142
144 POKE(J),Y:J=J+1:NEXTA
146 RETURN
165 GOTO 180
168 P.:INPUT"VERTICAL SHIFT(0-40)",V1:
    P.:INPUT"INDEPENDBNT VARIABLE",X:
    P.:INPUT"CHANGE IN X PER PLOT",X1:
    P.:INPUT"VARIABLE",A:P.:INPUT"VARIABLE",B:
    P.:INPUT"VARIABLE",C:P.:INPUT"VARIABLE",D:P.:H=1
170 Y =A*SIN(X)
171 Y=Y+V1
172 IF Y+6>ABS(46) GOTO 180
```

GRAPHICAL PLOTTER. (continued)

```
173 IF H+4>ABS(59) GOTO 180
174 IF Y+6<ABS(6) GOTO 180
175 SET(H+4,Y+7):H=H+1:X=X+X1:GOTO 170
180 LINE 23:P."KEY.Data,Graph,Axis,Plot,Instr"
185 INK.A1:IF A1=0 GOTO 185
186 IF A1=#47 GOTO 12
188 IF A1=#44 GOTO 168
190 IF A1 =#41 GOTO 126
191 IF A1=#50 GOTO 200
195 IF A1=#49 GOTO 1
198 GOTO180
200 INPUT"NUMBER OF PLOTS?",P:FOR Q=1TO P:P."PLOT NUMBER",Q:
    INPUT"CO-ORDINATE",X:P."PLOT NUMBER",Q:
    INPUT"CO-ORDINATE ",Y:SET(X+4,Y+7):NEXT Q
205 P."SEARCHING FOR PLOT"
210 Z=0
214 FOR X=4 TO 60:FOR Y=7 TO 47:IF POINT(X,Y) =-1 GOTO 220
216 GOTO 250
220 IF Z=0 GOTO 240
222 S=(Y-Y1)/(X-X1)
230 Y2=Y1+S*X2:SET(X1+X2,Y2):X2=X2+1:IF X1+X2=X GOTO 240
235 GOTO230
240 Y1=Y:X1=X:Z=Z+1:X2=0
242 IF ABS(Y)> 47 GOTO 180
244 IF ABS(Y)< 7 GOTO 180
246 IF X> 60 GOTO 180
248 IF Z=P GOTO 180
250 NEXTY:NEXTX
260 GOTO 180
280 STOP
```

CRYSTAL BASIC.

Crystal BASIC for INTERAK is the latest and most exciting addition to our software library. It is a 14K BASIC interpreter by Crystal Research with an implementation by the user group. It has all the usual instructions found in the best BASICs plus a few more such as RENUMBER, MUL$, SPEED, WIDTH. It has Multi-Dimension Arrays, Named Files on tape and five character variable names. I have listed the instruction set below. The cost of a tape and manual will be £40 from the user group. (This is less than the original from XTAL since we aren't registered for VAT). If you're interested please let me know as soon as possible so we can order the manuals.

P.V

Crystal BASIC:

ABS	:AND	:APPEND	:ASC	:ATN	:AUTO	:CALL	:CHAIN	:CHR$
CLEAR	:CLOSE	:CLS	:CONT	:COS	:CREATE	:DATA	:DEEK	:DEF
DEL	:DIM	:DOKE	:DRIVE	:ELSE	:END	:EVAL	:EXP	:FN
FMT	:FOR	:GOSUB	:GOTO	:HOLD	:HEX$	:IF	:INCH	:INP
INPUT	:INPUT#	:INT	:IOM	:KBD	:LEFT$	:LEN	:LET	:LIST
LN	:LOAD	:LOG	:MGE	:MID$	:MOD	:MON	:MUL$	:NEW
NEXT	:NOT	:NULL	:OFF	:ON	:OPEN	:OR	:OUT	:PEEK
PI	:POINT	:POKE	:POP	:POS	:PRINT	:PRINT#	:PTR	:READ
REM	:REN	:RENUM	:RESET	:RESTORE	:RETURN	:RIGHT$	:RND	:RUN
SAVE	:SCRN$	:SEP	:SET	:SGN	:SIN	:SIZE	:SPC	:SPEED
SQR	:STEEP	:STOP	:STR$	:SWAP	:TAB	:TAN	:TO	:THEN
VAL	:VERIFY	:WAIT	:WIDTH	:XOR	:ZONE	:@		

Programs in Law.

The next few pages contain a most interesting article from a member, clarifying some points of law concerning computer programs. It will provide reassurance to users who put off the idea of writing a good piece of software because they are frightened their work will be copied unfairly.

PROGRAMMES IN LAW

Much English law has ancient origins but has been and continues to be altered or interpreted to cover new developments.

Computers are new (at least on the legal clock which tends to work in microHertz), programs rather newer; so it won't come as a surprise that the law has not entirely come to terms with them.

Most readers will reckon they know what computer programs are but they can mean different things to different people - like loving, someone has said.

The fact that a program in itself is intangible gives the hidebound law a bit of a headache. It has, for instance, come to the conclusion that a program (as distinct from the medium on or in which it is embodied) cannot be stolen. This principle was established in the case of a student who got hold of the examination paper in advance, extracted the relevant information and returned it. He was found not guilty of theft.

So when it comes to the legal protection of programs you won't be surprised to hear that there are large areas in which instead of a "yes" or "no" answer you will be told "perhaps" "it is thought" "probably" and "it has yet to be decided".

There are three principal sectors of the law (all complex) which might be invoked to protect a program - Patents, Copyright and Confidentiality of Intellectual Property.

While it is just possible for an invention comprised in a computer program or, more probably, a computer so programmed to be the subject of a successful Patent Application the chances are small and the complications ans costs are enormous. So let's pass that one by!

Copyright is a fairly old legal concept dating back to the invention of the printing press but the law has undergone substantial development and is still developing. The position in this country is now governed by the Copyright Act 1956 - when computers were few and programs fewer.

This means that the Act makes no specific reference or provision for Computer Programs and they have to be considered in the context of provisions really designed to be applicable to rather different things. There are inevitably appreciable areas of uncertainty and so far as I am aware no case involving copyright in a computer program has come before the court for final decision or at any rate none has been reported.

Copyright is different in essence from a Patent which confers a virtual monopoly on its owner. Copyright is essentially the legal right to prevent one's own work (provided it derives from independent thought, skill or judgement) from being copied or reproduced by anyone else. The work involved must come within the definition of "a literary, dramatic or artistic work": a term which is liberally interpreted by the law.

Subject to the question of form - see later - there is no doubt that an original program, i.e. one evolved from first principles by the programmer, can be the subject of Copyright as can a program where the programmer has produced an original combination of sub-routines even though the latter individually could not be the subject of Copyright.

A problem which arises, in theory anyway, is whether a program which has no existance except on tape, disk or in ROM falls within the definition of a "work" for Copyright purposes.

In practice this probably matters little since most programs will from inception be embodied in some form of hard copy which would indubitably be sufficient to put matters beyond doubt.

It could be argued that the mere appearance of the program lines on a VDU screen in the course of its evolution would be sufficient to constitute a work for Copyright purposes.

The person entitled to the Copyright is normally the author except that where he is an employee and the "work" is within the sphere of his job the Copyright will normally belong to the employer unless some special contract exists. This is not the case for an independent free-lance or self-employed programmer or Company. In these cases in the absence of an agreement to the contrary the Copyright will belong to the actual author.

Next. - What constitutes an infringement of Copyright? - i.e. what can the author stop? Sect 2(5) of the 1956 Act sets these out as follows:-

(a) reproducing the work in any material form.
(b) publishing the work.
(c) performing the work in public.
(d) broadcasting the work.
(e) causing the work to be transmitted to subscribers to a Diffusion service.
(f) making an adaptation of the work.
(g) doing in relation to an adaptation of the work any of the acts specified in relation to the work itself in paragraphs (a) to (e) above.

These prohibitions apply also to any substantial (qualitative rather than quantitative) part of the work. Adaptation includes translation so that rewriting into a different language would constitute infringement.

Theoretically the mere <u>use</u> of a programme by itself does not constitute infringement but in practice it would be almost impossible to avoid a sufficient copying or reproduction to constitute an infringement.

The mere dealing with a program in which Copyright subsists for the purposes of research or private study is not an infringement but this defence is not normally available if associated with commercial exploitation.

The normal remedy for infringement is an injunction - an order of the Court restraining the infringer. Breach of an injunction is a Contempt of Court and with serious consequences for an infringer - prison or fine. The successful Plaintiff will also be entitled to damages for the infringement - generally the loss it has caused to him.

Most lawyers think that the Computer Industry has underestimated the effectiveness of Copyright law as a means of protecting rights in computer programs.

The main problem is usually detection but once an action for infringement has commenced the Plaintiff has a right to "discovery", i.e. effectively to see all relevant correspondence and documents which are or have been in the possession of the Defendant, with the further possibility in exeptional circumstances of an actual search of the Defendant's premises to prevent destruction of evidence.

Unlike Patents where the formalities are time consuming and expensive there are no formalities or registration required to establish Copyright. It attaches as soon as the work is created. If by chance two individuals were independently to produce exactly the same program each would be entitled to Copyright and could restrain everyone else except the other author. Obviously this won't happen in practice very often.

The above remarks apply to the United Kingdom, but most other developed countries have essentially similar legal provisions with local application. There are two relevant international conventions covering groups of Countries - the Berne Convention - principally European, and the more recent Universal Copyright Convention - actually not as wide as its name implies. Member countries agree to allow to foreign authors the same rights as their own citizens in Copyright material origination in another Member Country.

To secure protection in the ECC Countries it is necessary to incorporate in each copy of the material in which Copyright is claimed the symbol C, the name of the proprietor of the Copyright and the year of first publication.

The Confidence/Intellectual Property aspect must stand over for the time being.

This is a rather condensed outline of some points on the law of Copyright and its application to computer programs. For anyone who wants to delve deeper may I suggest Bryan Nibblett's "Legal Protection of Computer Programs" published by OYEZ Publications Limited?

Calamity Gulch (Our version of Silicon Valley)

VDU 2K Modification. ERRATA

User group Newsletter No. 2 page 17.

Find the paragraph that reads :-

1) Cut track U14 pin 3 - U22 pin 7 :Non-socket side. Ensure U22 pin 7 remains connected to U15 pin 1, best is to cut the track near to U14 pin 3 before it passes through the board.

Change the paragraph to now read :-

1) Cut track U14 pin 3 - U22 pin 7 :Non-socket side. Cut the track near to U14 pin 3 before it passes through the board.

On page 20 locate the "AFTER" drawing and then find the line that is drawn from U15 pin 1.

Erase (use a Tippex like fluid), the line from U15 pin 1 to where it joins the multiple branch dot that feeds U8 pin 12 and U22 pin 7. (Leave U8 pin 12 connected to U22 pin 7)

Draw a new line from U15 pin 1 to U8 pin 6.

The drawing should now show U8 pin 12 connected to U22 pin 7 and U15 pin 1 connected to U8 pin 6.

This completes the corrections.

I must apologise to everybody for the above error. It was an aberration which was well spotted by Chin Chinnery. Whom I must thank, not only for finding the error, but also for telling me so that I could take corrective action in time for this issue of the Newsletter.

Many thanks Chin.

Bob Eldridge. June 1983

P.S. To my knowledge four VDU-K cards have been modified and are working well!

Letter from Chris Davies:

Dear Peter,

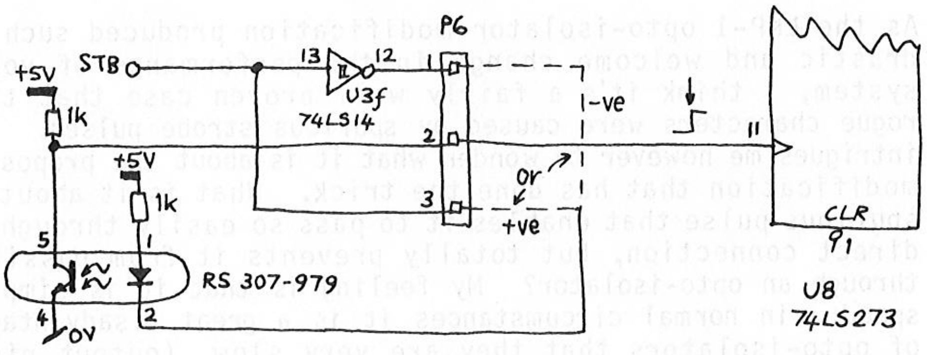
A belated thanks for supplying the games tape. It has taken me some time to sort out my tape system but at last I have got satisfactory results! My five year old son has had plenty of fun with Happy Sums. I don't know whether you intended to put more games on side 2 but I got a few from David Parkins to help out with my fault-finding session. For those who have had problems such as mine I offer a few words for Interaktion:-

<u>Some Hardware Ins and Outs</u>

For various reasons, financial or otherwise, some essential peripherals may not be quite what our leader had in mind.

First let's consider keyboards. Some of these are self-scanning types; often the data are continuously changing at the output even when no strobe is present. Those of us who haven't stuck to the good engineering practice of screening between conductors on the keyboard cable are asking for trouble. We get rubbish on the screen even before touching the keyboard! If you haven't any ribbon cable of prefer the appearance of your existing long multicore the following add-on to your LKP-1 card may be of interest. A standard opto-isolator is used to eliminate spurious pulses from the strobe input.

<u>Part of LKP-1</u>



This modification has been used successfully with RCA and ICL Keyboards. Even with eight feet of unscreened cable no spurious characters have been encountered. The components fit neatly to one end of the LKP-1 Patch area. No links are fitted to P6 but positive or negative strobes should be selected as necessary.

Now to move on from talkative keyboards and cables to quiet cassette recorders. My difficulty was in recovering my own recorded data without extremely critical adjustment of playback volume and tone controls. Azimuth adjustment was corrected, overall harmonic distortion, response and noise level were measured on sophisticated test gear. The audio results were more than adequate for a middle-of-the-range cassette recorder.

However, further tests pointed to some sort of tape problem so phase errors were considered. Investigation of the recorder input circuitry revealed some rather curious feedback arrangements. These resulted in a complex input impedance. It seemed worthwhile to try the most basic form of correction to alter the relative phase of the 1.2 kHz and 2.4 kHz tones. This took the form of a 10nF capacitor in series with the recorder input. A terrific improvement resulted, with no errors over a wide range of playback levels. Results were so good that no further optimisation was attempted and C11 in the DTI-1 card has been replaced with a 10nF component. It's well worth experimenting with different values for C11, C15 if you have similar tape problems, and much cheaper than buying a hi-fi cassette recorder!

Chris Davies

P.S. Copy sent to David Parkins for his info as I've taken up a lot of his time!

The Designer's Comments

It's easy to find problems, but much harder to solve them. Thank you Chris for a very positive contribution; if anyone is suffering these sort of difficulties I am sure they will be only too glad to try some experiments of their own along the lines you suggest. Some comments do occur to me however as follows:

1. As the LKP-1 opto-isolator modification produced such a drastic and welcome change in the performance of your system, I think it's a fairly well proven case that the rogue characters were caused by spurious strobe pulses. It intrigues me however to wonder what it is about the proposed modification that has done the trick. What is it about a spurious pulse that enables it to pass so easily through a direct connection, but totally prevents it from passing through an opto-isolator? My feeling is that it is simply speed: in normal circumstances it is a great disadvantage of opto-isolators that they are very slow, (output rise times in the order of several microseconds), but in this case it is quite an advantage - the fast (spurious) pulses don't get through but the relatively slow "official" strobe pulse does. I wonder if the same effect could be achieved by putting a simple series resistor and parallel capacitor into the strobe line?

   By the way, for production I think a little further work would be needed on the circuit. I haven't got the specification of the device suggested to hand, but I would guess the transmitter current with the values given to be in the order of 3 mA, and if a 20% current transfer ratio is speculated, then the output current is only 0.6 mA - according to me this is much less than the 5 mA or so which would be needed to guarantee a good logic "0" at the output of the opto-isolator with a 1k pull up resistor. Another tricky area is the output rise time. As a general recommendation, input clock rise and fall times for a device such as the 74LS273 should be no more than two hundred nanoseconds: the likely rise time of the opto-isolated circuit will be much longer, perhaps a few thousand nanoseconds.

2. The devastatingly simple DTI-1 circuit modification which has produced such miraculous results is of even greater interest. It is good advice to suggest experimenting with different values of capacitor for the purposes of phase compensation, as different recorders can be expected to have different input impedances (indeed different recorders may need different degrees of compensation).

   By the way don't think the suggested modification is a universal nostrum: I asked someone to try the suggested modification out on his system and it had quite the reverse effect: the 10nF capacitor introduced numerous errors!

   However, the results Chris quotes represent convincing evidence that some cassette recorders seem to give deliberate and severe phase distortion. It may be that the manufacturers are striving to give a better sound. (Apparently the human ear is insensitive to phase distortion and the curious feedback arrangements may be designed to optimise some particular parameter which does have an audible effect, with total disregard to the inaudible phase distortion). The art of "specmanship" is practised in audio equipment even more than in computers, and it is a sad fact that many people choose their audio equipment according to the printed specifications rather than what it sounds like - it is only commercial sense therefore for a manufacturer to optimise say the published frequency response of the recorder at the expense of other parameters, if the marketing men say that this is what sells recorders.

   There is an extremely interesting and relevant article on this subject on pages 20 - 24 of "Practical Electronics" for June 1983. It will be a little highbrow for many users, but it is worth reading since it shows there is much more to this subject than meets the eye. The article takes a different approach; it suggests a method of recovering data from a tape after it has been badly recorded - I think I prefer the idea of trying to prevent a bad recording being made in the first place! It is easy to see why tape is such a hit and miss affair on lesser machines than Interak - just on the DTI-1 tape interface card alone there are nearly 20 integrated circuits devoted to this all important task; this is more than some cheap computers have in total!

   Fortunately I have only met one other instance of a user having persistent difficulty in making tapes; everybody else I have spoken to is delighted with the interface, and says it is one of the strongest features of the Interak computer that they can make 100% reliable recordings even at the fastest baud rate.

I think it would be a very good idea to compile a list of the makes and models of cassette recorders users have, so that any which need special "tweaking" can be identified. To start us off, the models I have used are Sharp Stereo Cassette Decks Models RT-12 and RT-21 which both give perfect results.

David Parkins,
Greenbank Electronics.

Letter from R.Norton:

Dear Sir,

I now have my Interak computer up and running ZYMON/ZYBASIC/48K + SOUND and am quite pleased with the performance. I am a constructor cum bodger cum experimenter at heart, and Interak appealed to me because of the easy access to <u>all</u> signals etc., ease of mods, and availability of prototyping boards of workable size with decent edge connectors at reasonable prices. The only major irritation to date is my rather long keyboard cable accepting unwanted data from the freezer thermostat, but the solution is obvious. (Sell the freezer?) However, I feel that a couple of comments are in order where the correct solutions lay with your good selves.

The first, and probably most important point is that ZYMON does not allow the cassette interface to run at 2400 baud, and the ZYBASIC cassette is supplied at the faulty speed and takes around 2.5 minutes to load. When saved at 2400 baud it will load in around 45 seconds. This can be achieved by reducing the delay at ZYMON's address 047BH. I use 03 but have not checked the maximum value possible before interference starts with the cassette interface. There may be other solutions but that was the obvious one to me on inspecting the ZYMON listing.

My second point is more a plea to get that ZYBASIC manual to press as quickly as possible. The provisional manual is totally insufficient (for me anyway). My only in depth experience is with ZX81 BASIC, although I have played briefly with other computers' BASICs. My first problem, after keying in "Monster Mash" and saving it, was to find that there was no "LOAD" command in ZYBASIC, and it took nearly three hours of experimenting with every possible combination of letters that might be interpreted as a load command, before I stumbled across the truth. A small note in the manual about loading would at least let me know it hadn't been omitted accidentally, As yet I can't print to screen wherever I want without getting completely gummed up with "LINE", so am missing something somewhere. "FREE" doesn't seem to work, but is unecessary anyway. I even get lost with "LIST". "Monster Mash" is quite impressive for a BASIC program, so the BASIC works well for those who know it. Please more info ASAP.

I would like to add that the cassette interface is absolutely superb to use and I would like to compliment you on your documentation standards, which are in general, excellent.

Thank you for goods and assistance to date.

Yours gratefully,

R.Norton.

Reply to R.Norton's letter:

Thank you for the very valuable and constructive comments on the various points you have raised. Taking them roughly in order:

The recommended connection between the keyboard and cable is about 1 metre of 20-way ribbon cable. I have heard of a couple of instances of the trouble you describe but so far have not heard of trouble with the recommended cable and connections. The signals which most keyboards provide are normal "TTL" (As produced by Transistor-transistor logic ICs). Strictly speaking they should not be conducted for more than a few hundred millimetres without special twisted pair wire, special drivers and so on, because of the very fast voltage transitions which cause all manner of "ringing" and "reflections" (sorry I can't go into this subject more - shortage of brains more than shortage of space!). If ribbon cable is used and some care is taken about the signals on each side of the very vulnerable "strobe" signal a metre or so should cause no problems. The allocations are based on a commercial keyboard manufacturer's ("Alphameric Ltd."); we had arranged to supply it to Interak users, but the price suddenly shot up to impossible levels when one of Alphameric's major customers objected to the low price we were selling it.

If you write again, giving a detailed as possible description of the construction and allocations of the cable you use, power supplies, case, connectors and so on I will do my best to help with specific advice, but here are some general ideas and remarks which occur to me:

Let me stress that the behaviour you describe isn't normal, so we are looking for something very wrong somewhere.

1. Keep the keyboard lead as short as you can; use ribbon cable, with the standard connections - if any "scrambling" of the wires has to be done do it at the keyboard end of the wire. If you use your own wire, try to used a screened multicore

cable, and earth the screening. Earth any unused conductors in the cable, especially important if you have not used screened cable.

2. One possible experiment is to connect small (say 100p) capacitors from each signal line to ground on the LKP-1 card. Start with the strobe line as this is the one which is the most vulnerable to interference.

3. Make sure the SIL resistor packs on the LKP-1 have not been mixed up - they have two different values.

4. The computer should be mounted in an earthed metal case; if you don't have such a case, try a bit of earthed tinfoil to screen it from interference.

5. Use screened mains cable to be highly professional, and a mains interference suppressor; a most attractive type which is available combines the function of a chassis mounting plug and combined suppressor.

<u>Baud rate</u>: (Supercharger for ZYMON's Save Routine)

I don't want to sound terribly pedantic, but the 2400 (etc.) baud rates are fixed by the tape interface crystal and are quite definately 2400 (etc.). On for example a VDU terminal set to 9600 baud it is perfectly possible to type one character an hour, and still speak of it being transmitted at 9600 baud. The reason for the apparent reduction of the baud rate is that ZYMON 2 inserts a deliberate fixed "SCROLL" delay in between adjacent records, for a very good reason, (it is not "faulty"), to do with error checking, as we hope will be explained by the author of the ZYMON 2 program. The rate at which data is sent to the tape interface at 2400 baud with the "SCROLL" delay inserted is much less than the maximum possible without the delay, which explains the tremendous speed improvement when you alter its length.

However, like you I find the DTI-1 interface very reliable, and so I am never troubled by errors, thus the error checking is of only academic interest to me. My own experiments have resulted in the same improvements, and I propose in future that the ZYBASIC cassette be distributed without the delay. Instead of simply reducing the delay I removed it altogether by changing the following bytes in ZYMON 2 (any Interak 1 user can do this using ZYMON 2s "M" command, as ZYMON 2 runs in RAM):

Address	Existing	Change to
03B6	CDF204	000000
047C	CDF204	000000

With the above alterations I manage to get ZYBASIC loaded in about 33 seconds, which is very close to the theoretical minimum of 28 seconds at 2400 baud. (Calculated as follows: 6216 bytes to be recorded, x (1+2+8) since one start and two stop bits are added to the 8-bit bytes, divided by the baud rate (2400) = 28 seconds.)

Any users who try this alteration must remember that if ZYMON 2's tape save arrangements are "supercharged" in this way, any tape error on a load may prevent the subsequent part of the tape from loading correctly; as far as I am concerned (personal opinion only!) this is a small price to pay as I never get a tape error if I'm using good tape, and anyway I always keep more than one copy of something important. The vital thing to me is that the error checking routine still checks for and exposes errors, even at the fast rate, and I happily accept that it is now best to abort the rest of the load once an error has been found.

Your comments on the ZYBASIC manual are very fair; ZYBASIC has some useful features which only those in the know can use. I think somebody who uses it more frequently than I will be the best person to answer your specific points, but don't forget the "Bits of BASIC" series in the newsletter is specially written to cover the points which people ask about. A much larger comprehensive manual is being written, and when it is ready will be sent at no charge to all purchasers of the existing manual: I have seen some of the enlarged manual in draft form and am sure it will be very well received by all the ZYBASIC users.

Although I don't claim to be an expert on software, I think I have to disagree with your statement that FREE is unecessary: one use which occurs to me is to test that the array (which is stored in the free locations) is big enough to hold the number of items to be entered in the array. I am sure these points, and many more, will be covered in the eagerly awaited "big" ZYBASIC manual.

David Parkins,
Greenbank Electronics

I had an plea for help from a user the other day. He would probably prefer to remain nameless so I shall refer to him simply as S.D. His system was working fine until he moved house, and during the move it had mysteriously developed a fault. As always I was pleased to do what I could to help, but I was intrigued to hear of some additional background information. He said "My wife threw one of the boards across the road - she says it was an accident."

D.M.P.

More from the post-bag. (next page)

From Norman Moorcroft, MISTC FRAS, Astonomical, Aviation, and Space Writer.

He writes:

'I enjoyed our little chat very much.

Before the War I was Manchester/Liverpool Correspondent for a New York Musical Journal and visited Liverpool fairly frequently. Since I moved to the South-East in 1960 I have only been back in my native county (Lancashire) on one occasion (a funeral) - about ten years ago - and didn't recognise large parts of my native heath - Bolton! I actually left Bolton in 1951 but didn't move down here straight away.

I am still doing some free-lance writing but theoretically at least I am now retired - having passed my three score years and ten in 1981.

Hope to hear from you ere long.'

Needless to say, never one to miss an opportunity, I wrote back and suggested Mr. Moorcroft contribute an article or two to us, for free, just to keep his hand in!

D.M.P.

A new and useful little booklet has come into my hands. It is published by A.P.Carpenter, and is entitled 'Z-80 Programming Aid' (A5 size, about 20 pages). It sells for £1.50 + p & p, and is available from Greenbank Electronics. It has a complete list of all the Z80 op-codes, cross-referenced as Source to Object, with number of cycles, and also Object to Source.

D.M.P.

Copy to Pete Vella.

1? [illegible] 1983

Wallington Instruments

Wallington Instrument Company
Kimberley Place
Purley Surrey
CR2 2BX
Telephone
01-668 4315

Manufacturers of
Electrical and
Electronic Instruments
Proprietor: D.M. Horn

L0/2840
19 MAY 1983

Your Ref:

Our Ref: DMH/FMT

16th May 1983

Mr D M Parkins
Greenbank Electronics
92 New Chester Rd
New Ferry
Wirral
L62 5AG

Dear David

Many thanks for the VDU-K PCB now assembled and tested.

There seems to be a clanger on the Video Reverse Switch S3 in its Reverse Video (Open) position.

The video signal at 15/8 is positive for peak white and OV for black level, this is added to 21/11 via summing network R16,17,18.
As the sync is negative going it subtracts from the OV portion of Video (Black) level to give a negative going sync pulse. Normal standard for video signals is:-
Video 0.7V positive, Sync 0.3V negative.
With S3 open 21/8 signal inverts entirely to give the charac ter information at OV (Black) and blank area as peak white. This means that when sync is subtracted by adding a negative syns pulse, it is a sync pulse referred to white and not to black level and a normal video monitor will lose sync altogether.
My answer to it is to take 21/9 to the centre contact of a SPDT switch Select normal video display by taking one pole to earth (As per S3 as shown) but take the other pole (For reverse Video) to 20/5. This inverts the character display only giving a box display with black characters on a white background but now maintaining a proper sync output as the blank surround of screen is at black level.
The monitor now syncs properly and does not require any adjustment of Brilliance or contrast when changing from normal to reverse video.
For a better approximation to a normal video signal,suitable for a monitor with 1Vp-p into 75 ohms make R16 = 180 R18 = 470. This givesabout 0.6V video with -0.3V sync.

Yours sincerely

Don Horn

Don Horn

Reply to Don Horn's Letter:

Much as it pains me to say it, I fear you are perfectly correct regarding the "Invert All" setting of S3. The main mystery is not so much why the circuit does not work but why it does work in so many cases! The additional feature was not designed in, but was added at a very late stage (I think there are some remarks to that effect in the VDU-K Manual), when I noticed a conveniently spare gate lying around when the master artwork was being prepared. The "Invert All" switch was tried out on my prototype and worked O.K. so without a moment's further thought it was included permanently as an option. I am always preaching the futility of "designing with a soldering iron" and so it is poetic justice that I have been caught out making a mistake! Still, even Homer nods.

Your suggested modification will certainly get us off the hook, but quite a bit more will have to be done if we are to get the margins at peak white, yet get the sync levels right.

The video output was deliberately set at more than 1V peak to peak because we had heard of so many people (at the time) who needed a bit extra to suit their own, perhaps inferior, equipment, and who were being driven to mucking about with amplifiers, level shifters, and the like. Again, somewhere in the manual that point is already made, but I might have made life simpler for the majority if I had stuck to the straight and narrow instead of misguidedly trying to help the few.

Your letter is very much appreciated and is most constructive, in that you have suggested solutions as well as the problems! It also confirms that, in taking the trouble to write, you feel a part of the whole "Interak" project. From what I read in the technical press the users of many well-known computers feel totally alienated from the designers - it is very gratifying to hear from users who don't think of the relationship as "them and us" so much as "us and us".

David M. Parkins,
Greenbank Electronics.

FOR SALE

Mr. B.K.Jan, 17 Taylors Lane, London NW10 has various components for sale. UART, Memory, CPU etc. full spec. 'Phone user group for details.

Mr C.G.Evans, 18 Wilderswood Avenue, Horwich, Bolton has various own-design boards compatible with Interak - write direct for details.

USER GROUP BUYING POWER

A few people are a bit mystified when completing the section in the application form which asks whether or not they are interested in bulk buys with other users. The following "deal" for special prices which I have been offered for Interaktion group members is one example of how our combined buying power can encourage discounts from suppliers. All of the following are brand new. Contact me (Peter Vella) in the first instance. These prices were current when this was written - please note that they are very much subject to change.

Starprinter DP510:
100 characters per second 9 x 9, high resolution, 2.3K buffer. Epson code compatible. Centronics interface, RS-232 optional. Subscripts, superscripts, italic, four character sets. 2K user definable ROM. 8-10" single sheet, 3-10" sprocket feed, 8.5" roll paper, 0.5" x 2" inked nylon ribbon. 12 month return to depot warranty.

80 column version £270.25 including VAT.
132 column version £299.00 including VAT.

Juki Daisywheel Model 6100:
18 characters per second. RS-232 optional interface. Linear motor. 2K buffer. Diablo protocol, Wordstar compatible printing, 13" platen feed. (Cut sheet and tractor options) Triumph-Adler daisy wheel, IBM 82 single strike and multistrike ribbons.

Price £335.00 + VAT

Colour Printer/Plotter:
Of a similar specification to that offered for the "Oric" computer, this allows printing at up to 12 characters per second in up to four colours. 40/80 column printing, parallel interface, price includes 8 pens and paper: £130.00 + VAT.

12" Sanyo Monitors:
12" green screen 18 MHz bandwidth, composite video -ve sync. £91.50 + VAT.

12" white screen 15 MHz bandwidth, composite video -ve sync. £64.00 + VAT.

BMC 12" Monitor:
Green screen £91.50 + VAT.

P.V.

Software Library.

(Please enquire for cost of program / postage etc.)

NAME	DESCRIPTION	AUTHOR	CODE	SUPP.	FORMAT
ZYMON2	INTERAK monitor	BE	MC	GB	A,C
ZYBASIC2	INTERAK BASIC	--	MC	GB	A
XTAL BASIC	14K BASIC interpreter	XL	MC	UG	A,C
HC DISASS	Simple Disassembler	HC	MC	UG	A
ASM 32	Editor Assembler	--	MC	UG	A,C
VELTEXT	Text Editor	PV	MC	UG	A,C
RAKOVSKY	Chess	--	MC	UG	A,C
AC10.XX	(Chess Character EPROM for VDU-K)		--	~~GB~~ UG	A,C
AVALANCHE	Blob Dodging Game	DB	MC	UG	A
MONSTER MASH	Maze Game	BE	ZB2	UG	A
Graph	Graph Plotter	MC	ZB2	UG	A
Happy Sums	Fun maths	PV	ZB2	UG	A
Hangman	Spelling game	PV	ZB2	UG	A
O's and X's	Game	PV	ZB2	UG	A
Pools Pick	Random Draw Selector	PV	ZB2	UG	A
Count	Learn to count	PV	ZB2	UG	A
Dice Pontoon	Simple Game	PV	ZB2	UG	A

Key: MC machine code. ZB2 ZYBASIC. GB Greenbank. UG User Group. XL Crystal BASIC. Formats: A = 32 x 24 VDU-K, B = 64 x 16, C = 64 x 24 VDU-2K.

(Orders and enquiries to Interaktion User Group)

INTERAKTION BOOK LIBRARY.

This new section is to give members access to a wide range of books on computing and electronics. The only cost to the member is that of postage. Books may be borrowed for up to 3 weeks, and are available from the User Group address. I have managed to press-gang a member Dick Bowyer into acting as librarian for now. At present the books available are:

Title	Author/Publisher
LANGUAGE BOOKS	
TRS 80 Assembly Language Programming	Radio Shack
Z80 Assembly Language Programming Manual	Zilog
A Course in Basic Programming	Sinclair
Making the Most of your ZX 80	Tim Hartnell
30 Hour Basic	C.Prigmore
Basic for Home Computers a Self-Teaching Guide	B.Albrecht, L.Finkel & J.Brown
Course in Standard Coral 66	J.D.Halliwel & T.A Edwards
Simple Pascal	J.McGregor & A.Watt
Lecture Notes in Computer Science Pascal User Manual and Report	K.Jensen & N.Wirth
DATA BOOKS	
Mostek 1982/1983 Microelectronic Data Book (memory/CPU/Peripherals)	Mostek
Memory Data Book and Designers Guide 1980	Mostek
Bytewyde Memory Data Book 1981	Mostek
National Semiconductor Memory Data Book 1980	National
National Semiconductor Interface Data Book 1980	National
TTL Data Book	National

Title	Author/Publisher
DATA BOOKS (continued)	
The European Selection (memory/interface/linear)	Motorola
GENERAL & ELECTRONICS	
Computer Technology for Technicians and Technical Engineers Vol. 1	R. Watkin
Electronic Computers Made Simple	H. Jacobowitz
Test Instruments for Electronics (how to build test instruments)	M. Clifford
Practical Test Instruments You Can Build	W. Green
How to Troubleshoot & Repair Electronic Test Equipment	M. Horowitz
Computers and the Social Sciences	A. Brier & I. Robinson
MANUALS etc.	
Epson MX-80 Type II Operation Manual	Epson
Newbury 8000 Series VDU Terminal Operator Instruction Manual	Newbury Labs
Electronics Projects Index (A descriptive guide to 2500 projects published in popular magazines. Quite old now.)	Polytechnic

All books have been donated by users (a lot from Greenbank). If you have any books etc. surplus to requirements please let me have them.

Richard Bowyer

I N T E R A K T I O N   S U B S C R I P T I O N

Dear User,

Please use the form below to arrange payment of your membership fee.

Two tariffs apply:

1) Standing Order @ £6.00 per annum.

2) Cheque or Postal Order £7.00 per annum.

These fees include 4 issues of our mag and access to a technical book and software library.

Please make cheques etc. payable to INTERAKTION.

A Standing Order Mandate is to be found on a separate sheet for you to complete and send to your bank, if you want to take advantage of the reduced rate.

INTERAKTION Account held at	Midland Bank plc Stone Staffordshire
Account number 63036227	Sorting Code 40-43-14

I have today given instructions to my bank to transfer by Standing Order the sum of £6.00 per annum to the INTERAKTION account.

NAME .... Address ....
....
....
....
DATE ..../..../19... ....

Please complete the above and send it to:
P.P. Vella, 19 Ford Drive, Yarnfield, Nr. Stone, Staffs. ST15 0RP